VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Modul samoorganizačních map pro program Modeler neuronových sítí

Module Self-organizing Maps for Program Neural Net Modeler

2020 Bc. Jakub Komoráš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jakub Komoráš**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Modul samoorganizačních map pro program Modeler neuronových sítí
Module Self-organizing Maps for Program Neural Net Modeler

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je rozšířit existující program "Modeler Neuronových Sítí" o modul umožňující vytvářet a učit samoorganizační mapy a jejich vizualizace.

1. Nastudovat a popsat problematiku neuronových sítí a samoorganizačních map - Kohonenovy mapy.
2. Implementace modulu samoorganizačních map do programu "Modeler Neuronových Sítí" s možností volby různého typu sousedství.
3. Nastudovat a popsat problematiku paralelizace Kohonenových map.
4. Paralelizace Kohonenových map pro běh na distribuovaných výpočetních uzlech.
5. Experimenty a testování na zvolených datových sadách.

Práce bude obsahovat:
1. Přehled použitých technologií.
2. Implementaci výše popsané funkcionality.
3. Dokumentaci programového řešení s využitím diagramů jazyka UML.

Seznam doporučené odborné literatury:

[1] ROJAS, Raul. Neural networks: a systematic introduction. New York: Springer-Verlag, c1996. ISBN 3540605053.
[2] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. Deep learning. Cambridge, Massachusetts: The MIT Press, [2016]. ISBN 9780262035613.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. David Ježek, Ph.D.**

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 1. května 2020 .....

Rád bych na tomto místě poděkoval všem lidem, kteří mi byli nápomocni při tvorbě této práce. Převážně svému vedoucímu panu Ing. Davidu Ježkovi Ph.D. za pravidelné konzultace, zodpovědné vedení a věcné připomínky.

Abstrakt

Tato diplomová práce se zabývá rozšířením programu Modeler neuronových sítí, který je určený pro podporu výuky předmětu Neuronové sítě, o modul samoorganizačních map s možností volby různého typu sousedství. Tento modul umožňuje vytvářet, učit a vizualizovat vnitřní strukturu těchto sítí. Dále byl v rámci této práce implementován program pro paralelní běh samoorganizačních map na distribuovaných výpočetních uzlech. Práce vysvětluje základy fungování neuronových sítí se zaměřením na samoorganizační mapy. V další kapitole je popsána problematika paralelizace průběhu učení těchto sítí. Jedna z kapitol je věnována datovým sadám využívaných při testování neuronových sítí. Následují kapitoly popisující implementovaný modul a program pro paralelizaci učení. V poslední kapitole je provedena analýza a porovnání jednotlivých způsobů paralelizace a vlivu sousedství na průběh učení.

Klíčová slova: Neuronové sítě, Samoorganizační mapy, Kohonenovy mapy, Distribuované výpočty, Paralelismus

Abstract

This diploma thesis deals with the extension of the Neural Net Modeler program by a module of self-organizing maps with the possibility of choosing different types of neighborhoods. This program is used in teaching the subject Neural Networks. The module allows you to create, teach and visualize the internal structure of these networks. A program for parallel running of self-organizing maps on distributed computing nodes was also implemented. The work explains the basics of neural networks with a focus on self-organizing maps. The next chapter describes the issue of parallelization of the learning process of these networks. One chapter is devoted to the datasets used in the testing of neural networks. The following chapters describe the implemented module and the program for parallelization of learning. The last chapter analyzes and compares the various methods of parallelization and the influence of the neighborhood on the learning process.

Keywords: Neural networks, Self-organizing maps, Kohonens maps, Distributed computing, Parallelism

Obsah

Seznam použitých zkratek a symbolů

BMU	–	Vítězný neuron z anglického Best Matching Unit
BSOM	–	Dávková implementace samoorganizujících se map
DP	–	Datový paralelismus
MNS	–	Modeler neuronových sítí
NS	–	Neuronové sítě
ParallelSOM	–	Program pro paralelní učení samoorganizujících se map
SOM	–	Samoorganizující se mapy

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Neuronové sítě se v posledních letech staly velmi skloňovaným pojmem, ačkoliv se jejich vznik datuje do 40. let devatenáctého století, velký rozmach v širokém spektru oborů zažívají právě nyní. Výrobci mobilních telefonů se vzájemně předbíhají, kdo najde ve svém přístroji lepší využití těchto sítí, srdce chytré domácnosti se dle výrobců bez neuronových sítí neobejde a v neposlední řadě se jejich uplatnění najde i v autonomních automobilech.

Učení neuronových sítí je však velmi náročný proces, který může trvat i celé hodiny či dny, v závislosti na velikosti trénovacích dat a požadovaného stupně naučení. Z tohoto důvodu se učení neuronových sítí jeví jako velmi vhodný případ pro paralelizaci, tedy rozprostření výpočetní náročnosti učení mezi více procesorových jader nebo dokonce i mezi více výpočetních uzlů.

Aby bylo možné úspěšně dosáhnout cíle v podobě paralelziace učení, je v první řadě nutné nastudovat obecnou problematiku neuronových sítí. Přiblížením pojmu neuronové sítě se zabývá druhá kapitola této práce. Neuronové sítě, jak může být ze samotného názvu patrné, mají svoji předlohu v nervové soustavě živočichů. Cílem oboru neuronových sítí je co neblíže přiblížit funkci mozku živočichů. Za tímto účelem vznikl model umělého neurony, který je v různých podobách využíván jako základní jednotka umělých neuronových sítí. Existuje velké množství neuronových sítí, lišící se vzájemně počtem vrstev neuronů nebo způsobem adaptace.

Další kapitola detailně popisuje Kohonenovy mapy, které se jinak nazývají samoorganizující se mapy. Tento typ sítí je často aplikován na kategorizační problémy a cílem této práce je implementace paralelního učení pro tento typ sítí. Síť pracuje na principu hledání společných znaků mezi jednotlivými skupinami vzorů, kdy dochází k vytváření shluků neuronů, které představují jednotlivé kategorie. Výhodou Kohonenových map je využívání tzv. učení bez učitele, to znamená, že k trénovací množině není nutná druhá sada dat, která obsahuje očekávané výsledky.

Čtvrtá kapitola popisuje způsoby paralelní implementace učení neuronových sítí. Popsány jsou dva různé přístupy a to modelový a datový paralelismus. Dále jsou popsány konkrétní způsoby založené na datovém paralelismu, kdy dva jsou obecně využívané při paralelizaci neuronových sítí a dva jsou speciální případy navržené konkrétně pro Kohonenovy mapy.

Pro testování neuronových sítí jsou důležité před připravené datové sady, sloužící jako referenční body, jelikož je pro ně velmi dobře zdokumentované chování známých neuronových sítí. Těmto datovým sadám je věnována samostatná kapitola, ve které jsou detailně popsány 3 datové sady, které byly využity při vývoji a následném testování.

Jedním z cílů této diplomové práce je rozšíření již existujícího programu Modeler neuronových sítí o implementaci Kohonenových map. Tento program umožňuje vytváření a studování chování jednotlivých typů neuronových sítí pro potřeby výuky studentů. Kapitola popisuje jednotlivé funkce programu dostupné pro samoorganizující se mapy a rozebírá implementační detaily.

S programem Modeler neuronových sítí úzce spolupracuje také program pro paralelní učení Kohonenových map, který je druhým cílem této práce. Tento program je určen pro distribuci výpočtů prováděných při učení Kohonenových map mezi velký počet výpočetních uzlů s cílem

minimalizovat dobu potřebnou k naučení sítě. Funkce a detaily programu popisuje kapitola sedm.

Osmá kapitola analyzuje jednotlivé přístupy paralelizace. Testy jsou prováděny na superpočítači s mnoha výpočetními uzly, tedy v prostředí, pro které je program primárně určen. V rámci Kohonenových map hraje důležitou roli tzv. funkce sousedství, proto je v kapitole také zkoumán vliv typu sousedství na průběh učení. Na základě těchto experimentů by mělo být učiněno doporučení pro vhodný způsob paralelizace využitý při učení samoorganizačních se map.

2 Neuronové sítě

Ačkoliv by se mohlo zdát, že neuronové sítě, někdy uváděny také jako umělé neuronové sítě, jsou mladým oborem, není tomu tak. Za počátek tohoto oboru se považuje práce „A logical calculus of the ideas immanent in nervous activity“ autorů Warrena McCullocha a Waltera Pittse [14] z roku 1943. V této práci navrhli jednoduchou matematickou reprezentaci základní buňky nervové soustavy, neuronu. Parametry tohoto neuronu byly většinou bipolární, nabývaly hodnot -1, 0 a 1. Ve své práci také ukázali, že pomocí NS založených na tomto jednoduchém neuronu je možné počítat libovolné aritmetické a logické funkce.

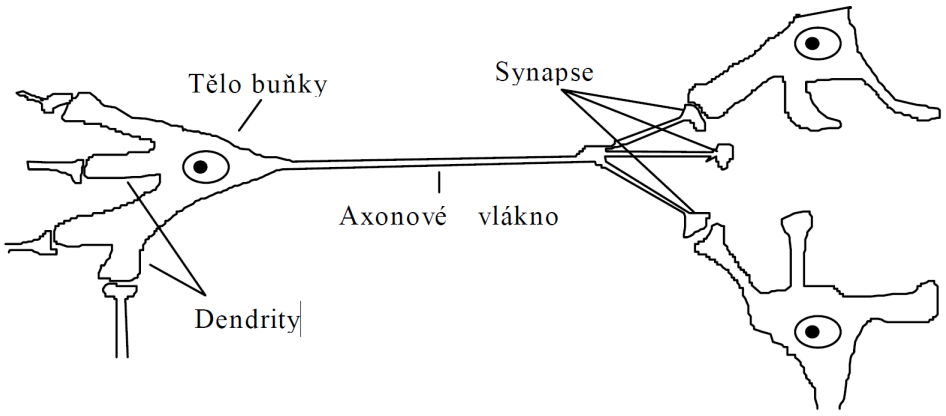
Na práci McCullocha a Pittse navázalo mnoho autorů se svými návrhy NS či počítačů inspirovaných neuronovou soustavou živočichů. Žádný z těchto návrhů však nepřinesl výrazné úspěchy. V roce 1949 přišel Donald Hebb ve své knize s myšlenkou učícího algoritmu pro NS. Zobecněním modelu neuronu z roku 1943 pro parametry z reálného číselného oboru vznikl tzv. perceptron. S tímto návrhem přišel v roce 1957 Frank Rosenbatt. Pro perceptron vymyslel také způsob učení a dokázal, že pro danou trénovací množinu existuje konečný počet kroků, po kterém je nalezen váhový vektor parametrů, přičemž nezáleží na počátečních podmínkách. Na výzkumu perceptronu se také zakládal první úspěšný neuropočítač. Zaměřený byl na rozpoznávání a klasifikaci znaků abecedy. Vstupem bylo pole 400 foto-vodičů a váhy mezi neurony představovalo 512 potenciometrů.

Na takto položených základech pro obor NS pokračovalo mnoho dalších vědců ve snaze, co nejreálněji simulovat funkčnost lidského mozku a nervové soustavy [24].

2.1 Biologický versus umělý neuron

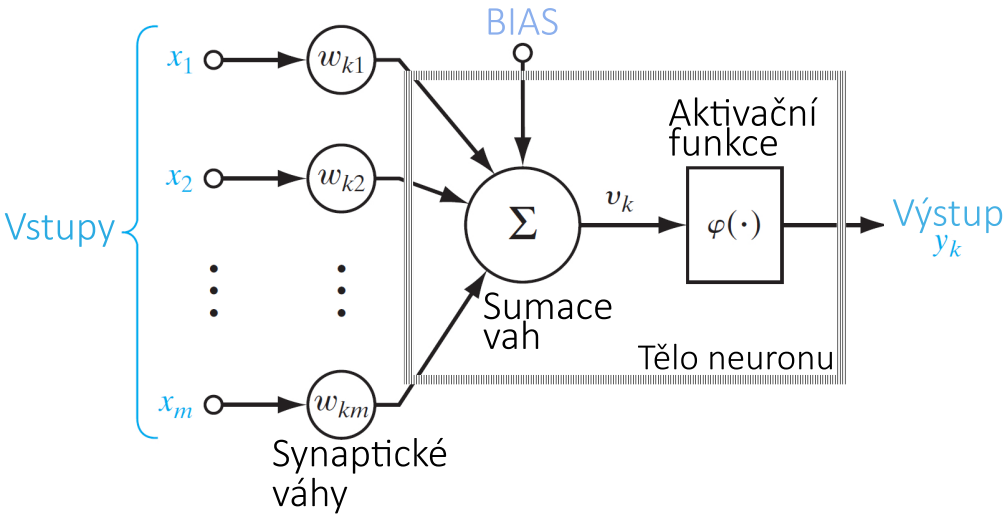
Inspiraci pro vznik prvotního matematického modelu neuronu nalezneme v přírodě a to v nervové soustavě živočichů. Nervová soustava živočichů se skládá z jednoduchých buněk, neuronů, které jsou vzájemně spojeny a vytvářejí sítě. Struktura a síla těchto spojů, neboli vazeb, určuje uložené znalosti v mozku. Mozek má schopnost vytvářet či rušit propojení mezi neurony, čímž se dokáže adaptovat na různé situace potřebné ke zpracování informací [10].

V lidském mozku nalezneme přibližně 10^{11} propojených neuronů. Komunikace mezi neurony je zajištěna v podobě elektrických signálů. Spojení neuronů je zprostředkováno pomocí synapsí. Synapse se nachází v místě styku axonu jednoho neuronu s dendritem neuronu druhého. Strukturu biologického neuronu a propojení s ostatními neurony můžeme vidět na obrázku 1. Každý neuron může přijímat signály z mnoha tisíc neuronů. Tyto signály jsou v těle neuronu vyhodnoceny a pokud překročí určitou prahovou hodnotu, dojde k tzv. excitaci neuronu. Excitace znamená, že neuron vygeneruje impuls, který je prostřednictvím axonu přenesen na další neurony. Synapse lze dle účinku na daný neuron dělit na inhibiční a excitační. Inhibiční synapse utlumují schopnost excitace a brání vygenerování impulsu, zatímco excitační synapse působí přesně opačně [9].



Obrázek 1: Biologický neuron [24]

Stejně jako je tomu u neuronových soustav živočichů, tak i u umělých NS je základní stavební a procesní jednotkou neuron. Schéma umělého neuronu je znázorněno na obrázku 2. I umělý neuron se skládá ze tří základních částí. Vstupy $x_1 ... x_m$, které odpovídají dendritům biologického neuronu, práh pro excitaci představuje aktivační funkce a axonu odpovídá výstup y_k. Pro každý vstup je specifikována váha, například pro vstup x_1 s označením w_{k1}, která charakterizuje vliv daného vstupu na další chování neuronu. Váhy mohou nabývat kladných i záporných hodnot, tak aby bylo možné simulovat inhibiční i excitační účinky [10].



Obrázek 2: Umělý neuron [10]

2.1.1 Excitace a výstup umělého neuronu

Reakce neuronu na sadu aktuálních vstupů se děje ve dvou krocích. Krok jedna je sumace vstupů, každý ohodnocený svou váhou. Tento krok znázorňuje rovnice 1. Výsledek, označme v_k,

se nazývá vnitřní potenciál neuronu.

$$v_k = \sum_{j=1}^{m} w_{kj} x_j \tag{1}$$

Do sumace vstupů často vstupuje také bias. Bias napomáhá ke zvýšení či snížení vnitřního potenciálu v_k. Ve výpočtu se s hodnotou bias pracuje jako s dalším vstupem ohodnoceným váhou, můžeme označit jako $x_{k0} = 1$ a w_{k0}. Výpočet vnitřního potenciálu s hodnotou bias znázorňuje rovnice 2.

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{2}$$

Dalším krokem je ohodnocení aktivační funkce, kde jako vstup použijeme vnitřní potenciál neuronu v_k. Aktivační funkce,někdy označovaná také jako přenosová funkce [26], limituje výstup neuronu. Typicky se výstup neuronu normalizuje na rozsah intervalu [0,1], nebo [-1,1] [10]. Nejčastěji používané funkce jsou

- perceptron,
- binární funkce,
- logistická funkce, neboli sigmoida,
- hyperbolický tangens [26].

Jako aktivační funkce může být například použita také funkce signum, dle rovnice 3, která normalizuje výstup neuronu na interval [-1,1] [10].

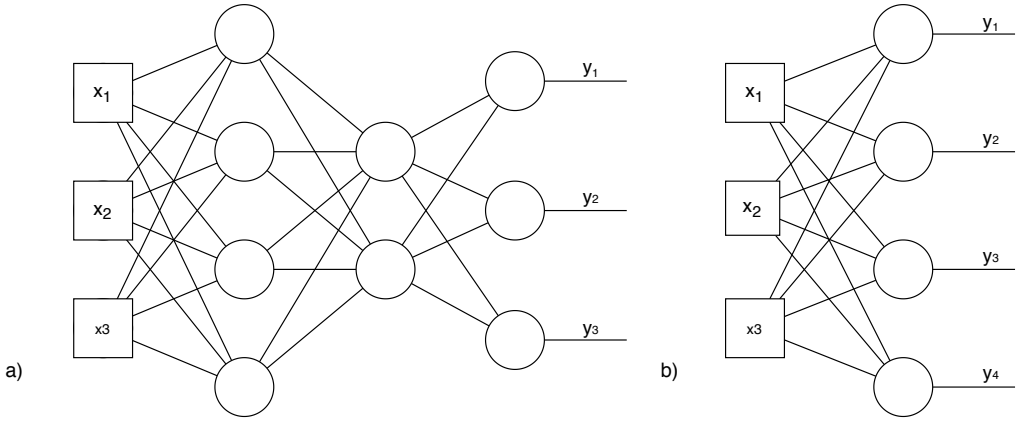
$$y_k = \begin{cases} 1, & \text{pokud } v_k > 0 \\ 0, & \text{pokud } v_k = 0 \\ -1, & \text{pokud } v_k < 0 \end{cases} \tag{3}$$

2.2 Dělení neuronových sítí

Neuronové sítě můžeme dělit podle několika vlastností. I přesto, že je vždy základní jednotkou NS neuron, mohou se sítě lišit použitou aktivační funkcí, propojením mezi neurony či učícím algoritmem.

2.2.1 Počet vrstev

První způsob dělení je podle počtu vrstev. Jednou skupinou v rámci toho dělení jsou sítě jednovrstvé nebo dvouvrstvé. Patří zde Hopfieldova a Kohonenova síť. Tyto sítě mají většinou specifikován svůj vlastní učící algoritmus a topologii. Příklad jak může vypadat uspořádání Kohonenovy sítě je znázorněn na obrázku 3b. Druhou skupinou jsou sítě se třemi a více vrstvami.



Obrázek 3: Struktury neuronových sítí dle počtu vrstev

V této kategorii se nejčastěji využívají sítě k učení algoritmus Backpropagation a liší se pouze počtem a topologií jednotlivých vrstev. Možné uspořádání více vrstvé sítě je vidět na obrázku 3a. Spojení neuronů je většinou uskutečněno tak, že výstup neuronu z nižší vrstvy je propojen se vstupy všech neuronů ve vrstvě následující. Každý s těchto propojů je ohodnocen váhami, které vyjadřují důležitost spoje pro daný neuron.

2.2.2 Algoritmus učení

Další způsob dělení je dle způsobu jakým probíhá učení. Zde rozlišujeme dvě možnosti a to učení s učitelem nebo učení bez učitele. Učení s učitelem si můžeme představit jako snahu sítě se svým výstupem přiblížit očekávanému výstupu pro daný vstup. Příkladem této kategorie jsou sítě využívající algoritmus Backpropagation. Pro tento typ učení se musí trénovací množina skládat ze vstupu a očekávaného výstupu.

Oproti tomu při učení bez učitele se síť snaží vycházet pouze z informací, které jsou součástí vstupních dat. Neuronová síť při tomto způsobu učení sama analyzuje vstupy, uspořádává je například do shluků na základě podobných vlastností. Tento způsob učení se také nazývá samoorganizace a do této kategorie spadají například Kohonenovy mapy, které jsou detailněji rozebrány v následující kapitole.

2.2.3 Styl učení

Poslední způsob jakým můžeme rozdělit NS je na základě stylu učení. Stylem učení rozumíme, jakým způsobem jsou v rámci sítě nastavovány jednotlivé váhy. Pokud je ke zjištění a adaptaci vah použit přesně daný výpočet, jedná se o deterministický styl učení.

Druhou možností je stochastický styl učení. V tomto případě jsou váhy nastaveny na základě náhodných čísel. Stochastický styl se většinou využívá pro inicializaci vah na začátku učení, které dále pokračuje deterministickým způsobem [26].

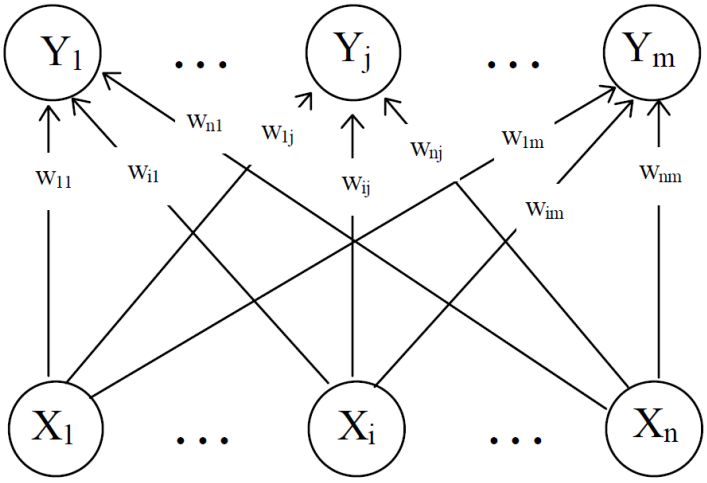
3 Kohonenovy mapy

Kohonenovy mapy patří mezi jedny ze základních typů neuronových sítí. Tyto sítě se také jinak nazývají samoorganizující se mapy, anglicky Self-Organizing Maps, zkráceně SOM. Jak již název samoorganizující napovídá jedná se o neuronové sítě s učením bez učitele. Pro tyto sítě je snadnější získat tréninkovou množinu dat, jelikož k nim nepotřebujeme druhou sadu dat v podobě očekávaných výsledků.

Síť pracuje na principu zjišťování podobnosti mezi vstupními daty, kdy hledá společné znaky a odlišnost, podle kterých se pak rozhoduje ve své aktivační fázi. Na základě této podobnosti dochází ke shlukování vstupů se společnými znaky do skupin, a proto jsou Kohonenovy mapy často aplikovány v prostředí, kde je třeba rozlišovat či třídit různé objekty [23]. Často se Kohonenovy mapy využívají na převod mluveného slova do psaného textu [5].

Příklady reálného použití Kohonenových map:

- detekce objektů na fotografiích,
- zpracování řeči na psaný text,
- zpracování obrazu,
- hledání podobností v neznámých signálech [23].



Obrázek 4: Obecné schéma Kohonenovy mapy [24]

3.1 Struktura sítě

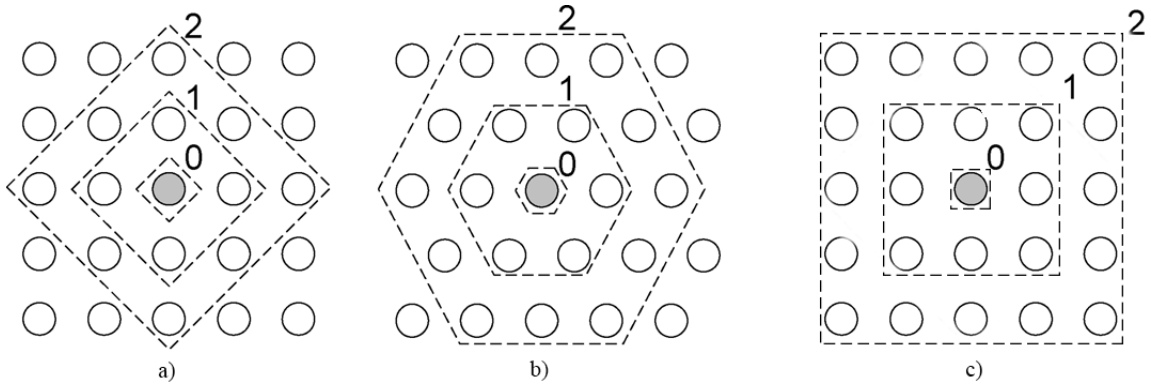
Kohonenovy mapy jsou dvouvrstvé sítě, kde jsou všechny neurony z první vrstvy propojeny se všemi neurony s druhé vrstvy. Schéma obecné Kohonenovy mapy lze vidět na obrázku 4, kde

jsou neurony první, neboli vstupní vrstvy, značeny $X_1 ... X_n$ a neurony druhé, neboli výstupní vrstvy, značeny $Y_1 ... Y_m$.

Druhá, výstupní vrstva, může být uspořádána do různých topologických struktur. Uspořádání těchto topologických struktur ovlivňuje proces adaptace, neboli učení, jelikož určuje, které neurony jsou považovány za sousedy. V rámci učení se totiž definuje pojem sousedství J výstupního neuronu j^* s poloměrem R. Sousedství lze definovat jako podmnožinu všech neuronů, jejichž vzdálenost je od neuronu $j*$ menší nebo rovna R. Matematický zápis viz rovnice 4, kde právě vzdálenost neuronů $d(j, j^*)$ je dána topologickým uspořádáním neuronů ve výstupní vrstvě [24].

$$J = \{j; d(j, j^*) \leq R\} \tag{4}$$

Jako nejčastější uspořádání výstupní vrstvy se používá jedno nebo dvourozměrná mřížka. Na obrázku 5 lze vidět typy možných sousedství pro dvourozměrné uspořádání výstupní vrstvy, hodnota vedle hranic sousedství udává jeho poloměr R.



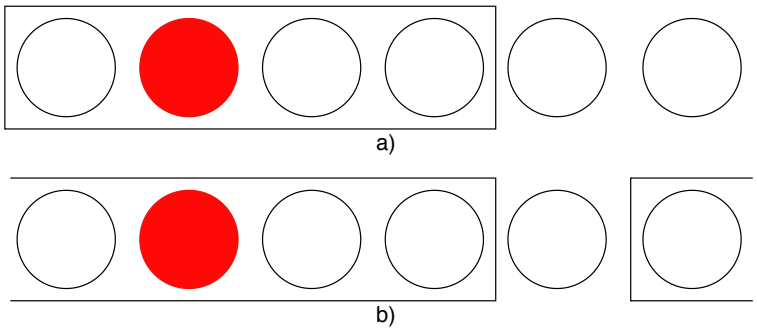
Obrázek 5: Typy sousedství při dvourozměrném uspořádání výstupní vrstvy [2]

Z obrázku 5 můžeme také vidět, že typ sousedství má zásadní vliv na počet sousedů. Počty sousedů pro $R = 1$ v závislosti na typu sousedství jsou vypsány v tabulce 1 [2].

Tabulka 1: Počet sousedů pro různé typy sousedství pro $R = 1$

Typ sousedství	Počet sousedů
čtvercové, obrázek 5a	4
hexagonální, obrázek 5b	6
čtvercové, obrázek 5c	8

Kromě tvaru lze sousedství specifikovat také způsobem chování na konci mřížky, tedy zdali lze za sousedy považovat i neurony na protější straně mapy či nikoliv. Toto chování je pro jednorozměrnou mřížku zobrazeno na obrázku 6. Na obrázku 6a je sousedství ukončeno koncem mřížky, zatímco na 6b sousedství pokračuje na protější stranu [18].



Obrázek 6: Způsob chování na konci mřížky pro R = 2

3.2 Průběh učení

Učení Kohonenovy mapy probíhá iterativně. Postupně dochází k procházení jednotlivých vstupů v tréninkové množině, kdy po předložení jednoho vzoru probíhá mezi neurony sítě soutěžení o vítězný neuron. Vítězný neuron je vybrán na základě podobnosti s předloženým vstupem, proto je někdy označován jako BMU z anglického best matching unit. Pro BMU jsou určeny neurony spadající do jeho sousedství. U BMU a jeho sousedů jsou aktualizovány jejich váhy spojů se vstupní vrstvou.

Sílu změny vah určuje parametr učení α, který nabývá hodnot od 0 do 1. Parametr α nabývá na začátku procesu hodnotu blízkou 1 a v průběhu učení je postupně snižován až na téměř nulovou hodnotu. Snížení parametru učení pod určitou mez může být jednou z možných podmínek ukončení.

Druhým parametrem použitým v rámci procesu učení je poloměr sousedství R. I u parametru R dochází během učení k postupnému snižování k nulové hodnotě. Na začátku je parametr R nastaven například na polovinu velikosti výstupní vrstvy [24].

Průběh učení lze popsat následujícími kroky:

a) Inicializace

V tomto kroku se inicializují jednotlivé váhy spojů mezi vstupní a výstupní vrstvou, obvykle na malá náhodná čísla. Dále je dochází k nastavení parametru učení α a poloměru sousedství R.

b) Předkládání vzorů z trénovací množiny

V tomto kroku dochází k postupnému procházení vstupů z trénovací množiny a pro každý vstupní vektor $x = (x_1, ..., x_n)$ se provede následující:

1. Pro každý neuron J s indexem j ve výstupní vrstvě se vypočte vzdálenost od vstupního vektoru pomocí rovnice 5.

$$D(j) = \sum_{i}^{n} (w_{ij} - x_i)^2 \tag{5}$$

2. Najdeme BMU, tedy ten s nejmenší vzdáleností $D(j)$ a jeho sousedy dle rovnice 4.
3. Pro všechny sousedy a BMU aktualizujeme váhy mezi nimi a vstupní vrstvou, na základě rovnice 6.

$$w_{ij}^{new} = w_{ij}^{old} + \alpha(x_i - w_{ij}^{old}) \tag{6}$$

4. Aktualizace parametru α a R.

c) Kontrola ukončovací podmínky

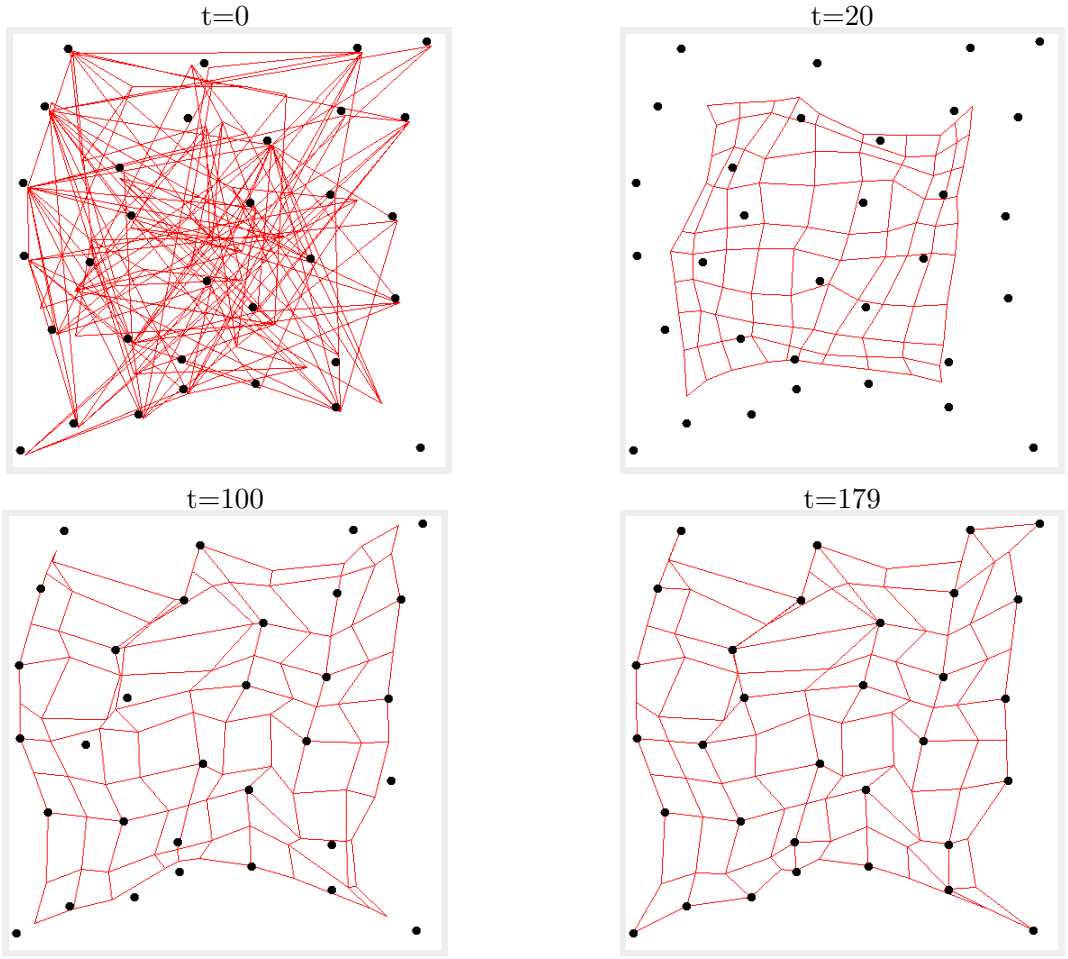
Pokud jsou splněny ukončovací podmínky je fáze učení ukončena. Pokud nejsou splněny podmínky pro ukončení opakuje se krok b) [24].

Na výpise 1 je popsáno učení SOM pomocí pseudokódu. V tomto případě je jako ukončovací podmínka zvoleno, že učení probíhá dokud je parametr učení α větší než hodnota 0,01.

```
initWeights()
initParameters()
while(α > 0.01)
{
      foreach(x = (x_1, ..., x_n) in trainset)
      {
        find j* where d(x, j*) = min{d(x, j) | j ∈ Neurons}
        w_{j*}^{new} = w_{j*}^{old} + α(x − w_{j*}^{old})
        for all j = {d(j, j*) ≤ R} do
        {
            w_j^{new} = w_j^{old} + α(x − w_j^{old})
        }
      }
 updateParameters()
}
```

Výpis 1: Průběh učení SOM

Průběh učení si lze také představit jako snahu sítě o nalezení prostorové reprezentace složitějších datových modelů. Podobné vstupy těchto modelů jsou poté reprezentovány neurony blízkými v rámci topologie výstupní vrstvy. Tuto podobnost je možné pozorovat i v případě lidského mozku.

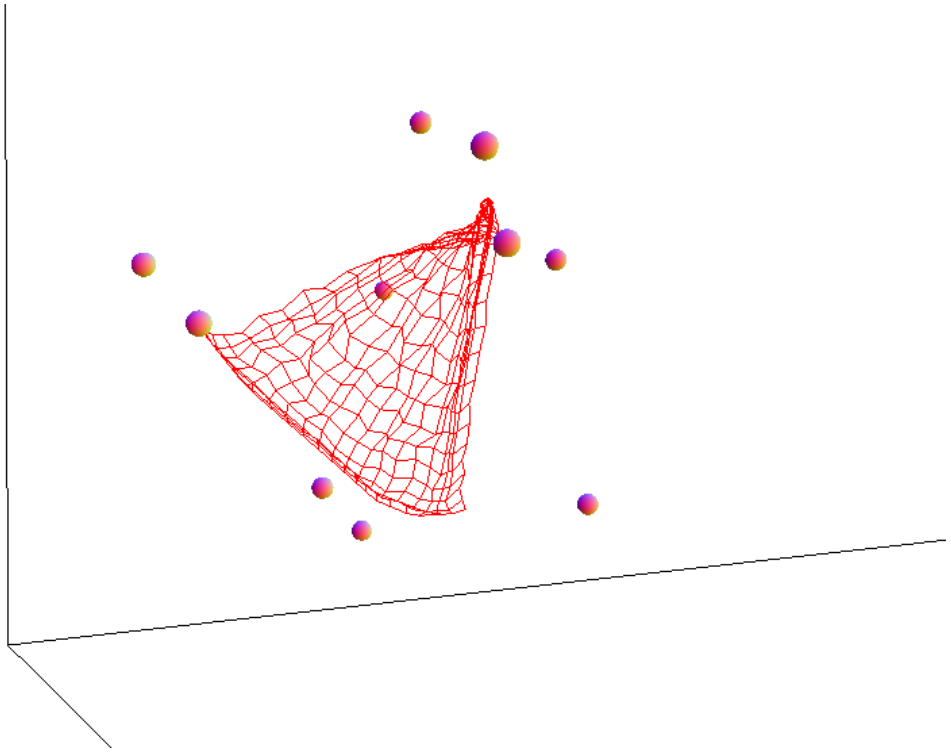


Obrázek 7: Proces pokrývání vstupního prostoru výstupní vrstvou sítě SOM

Pokud jako vstupní datový model vezmeme ohraničený dvou dimenzionální prostor, kdy jednotlivé vstupy jsou reprezentovány jako body se dvěma souřadnicemi, bude se během procesu učení snažit výstupní vrstva neuronů pokrýt právě tento prostor. Obrázek 7 zobrazuje průběh adaptace právě pro tento případ. Černé body na obrázku představují trénovací množinu vstupů a červené přímky spojují vždy dva sousedící neurony. Na počátku jsou váhy nastaveny náhodně na hodnoty ze vstupního prostoru. Dále dochází k postupnému zformování výstupní vrstvy a poté k pokrytí celého vstupního prostoru [25]. Obrázek 8, který byl pořízen z programu MNS, ukazuje, jak může vypadat struktura sítě během učení v trojrozměrném prostoru, trénovací množinu představují barevné kuličky.

3.3 Vyvolání informace

Po dokončení učící fáze se jako vstup používá neznámý vektor hodnot z oboru stejného jako je obor vektorů v trénovací množině. Stejně jako v učící fázi se po předložení vstupního vektoru



Obrázek 8: Zobrazení strukutry sítě v trojrozměrném prostoru, t=20

vyhledá BMU. Tento neuron zastupuje definovanou skupinu, která je neznámému vstupnímu vektoru nejbližší a tímto je neznámý vektor zařazen do již známé skupiny.

Při vyvolání informace lze použít dva režimy. Neadaptační režim odpovídá způsobu popsanému výše, kdy dojde pouze k zařazení neznámého vstupu do známé kategorie. Druhým režimem je adaptační režim, kdy první část je stejná jako u neadaptačního režimu, avšak zároveň dochází i k malé aktualizaci vah pro vítězný neuron. Tím je způsobené neustálé učení a síť je schopná se postupně adaptovat na dlouhodobě působící změny [23].

4 Paralelizace Kohonenových map

Neuronové sítě se staly velmi účinné pro řadu úkolů, například klasifikaci obrazů či rozpoznávání řeči. Velký pokrok umožnil nárůstu výkonu hardware, díky kterému bylo možné vytvářet větší modely NS nebo použít mnohem větší datové sady k tréninku těchto NS. I přesto je však stále čas limitujícím kritériem pro mnohé aplikace NS. Proto každé zlepšení vedoucí k redukci času potřebného k učení, může pomoci mnoha oblastem pro aplikaci NS. Díky tomu bude možné zvětšit samotný model NS, zlepšit kvalitu naučení NS či rozšířit množinu dat pro trénink. Jedním z možných způsobů, jak docílit rychlejšího učení, je paralelizace. Dva hlavní způsoby paralelizace jsou datový a modelový paralelizmus. Pro rozsáhlé modely NS je doporučeno tyto způsoby kombinovat, aby bylo možné dosáhnout co nejlepších výsledků [20].

4.1 Modelový paralelizmus

Při využití modelového paralelizmu dochází k rozdělení samotného modelu NS na menší části, kde každá je schopna běžet na odlišném výpočetním uzlu. Všechny uzly dohromady tedy tvoří jednu NS. Rozdělení může být provedeno například na základě jednotlivých vrstev u vícevrstvých NS [3].

U SOM může být modelový paralelismus proveden rozdělením výstupní vrstvy na menší části, kdy za každou část odpovídá jiný výpočetní uzel. Při předložení trénovacího vstupu je na menších částech rychlejší vyhledání lokálních BMU, ze kterých je následně vybrán jeden globální BMU. Výhodou je zachování stejného způsobu aktualizace vah jako u sekvenční verze, a proto tento způsob produkuje stejné výsledky. Velkou nevýhodou tohoto způsobu je však častá komunikace mezi uzly, která probíhá každou iteraci při hledání globálního vítězného neuronu. Tato komunikace limituje možnosti škálování a přináší zpoždění při zpracování každého vstupu [21].

4.2 Datový paralelizmus

Datový paralelismus je velmi populárním způsobem. V rámci datového paralelismu je na každém výpočetním uzlu spuštěn plnohodnotný model NS. Všechny modely jsou na začátku identické. Poté je rozdělena vstupní trénovací množina a části jsou rozdistribuovány mezi jednotlivé výpočetní uzly. Pro tyto části dat proběhne na každém uzlu proces učení a následně jsou aktualizované váhy synchronizovány mezi všemi uzly. Pokud je to nutné, proces se může opakovat, dokud není například splněn požadovaný stupeň naučení či jiná ukončovací podmínka.

Komunikace je v tomto případě redukována pouze na synchronizaci vah mezi uzly. Tato synchronizace však probíhá pouze po kompletním zpracováním části vstupní množiny na každém uzlu [3].

4.3 Dávková SOM

Dva dříve zmíněné způsoby paralelizace je obecně možné aplikovat na libovolný typ NS. V této práci se však zabývám paralelizací SOM, proto zde uvedu ještě dva způsoby, určené přímo pro tento typ NS. Způsob učení, které popisuje kapitola o Kohonenových mapách, je někdy označován jako on-line učení, protože k aktualizaci vah dochází po předložení jednotlivých vzorů. Oproti tomu u dávkového učení jsou váhy aktualizovány až na konci každé epochy.

Po předložení jednoho vzoru je stejně jako u on-line verze vyhledán BMU a jeho sousedi. V průběhu epochy jsou pro BMU a sousedy postupně aktualizovány jednotlivé sumy v rovnici 7. Dle rovnice 7 jsou vypočteny nové váhy na konci každé epochy. Funkce sousedství $h_{cj}(t)$ zde představuje míru ovlivnění daného neuronu daným vstupem.

$$w_{ij}^{new} = \frac{\sum_{t=t_0}^{t=t_f} h_{cj}(t) x_i(t)}{\sum_{t=t_0}^{t=t_f} h_{cj}(t)} \tag{7}$$

Tento přístup má pár výhod oproti on-line učení. Jelikož nejsou váhy aktualizovány po každém předloženém vzoru, ale až na konci epochy, nezáleží na pořadí předkládání vzorů. Dále dávkové učení nevyužívá učící koeficient α, tudíž odpadá potencionální možnost jeho špatného nastavení [13].

4.4 GM-SOM

GM-SOM, celým název Global-Merged SOM, vychází také z datového paralelizmu, kdy je možné proces popsat následujícími kroky.

a) Rozdělení trénovací množiny

V tomto kroku dojde k rozdělení vstupní trénovací množiny na zadaný počet částí. Přesnost této metody stoupá s počtem částí, avšak autoři doporučují volit počet částí dle rovnice 8, kde k je počet vzorů v trénovací množině, N označuje celkový počet neuronů ve výstupní vrstvě a p je zvolený počet částí. S počtem zvolených částí totiž roste také velikost trénovací množiny použité v poslední části algoritmu.

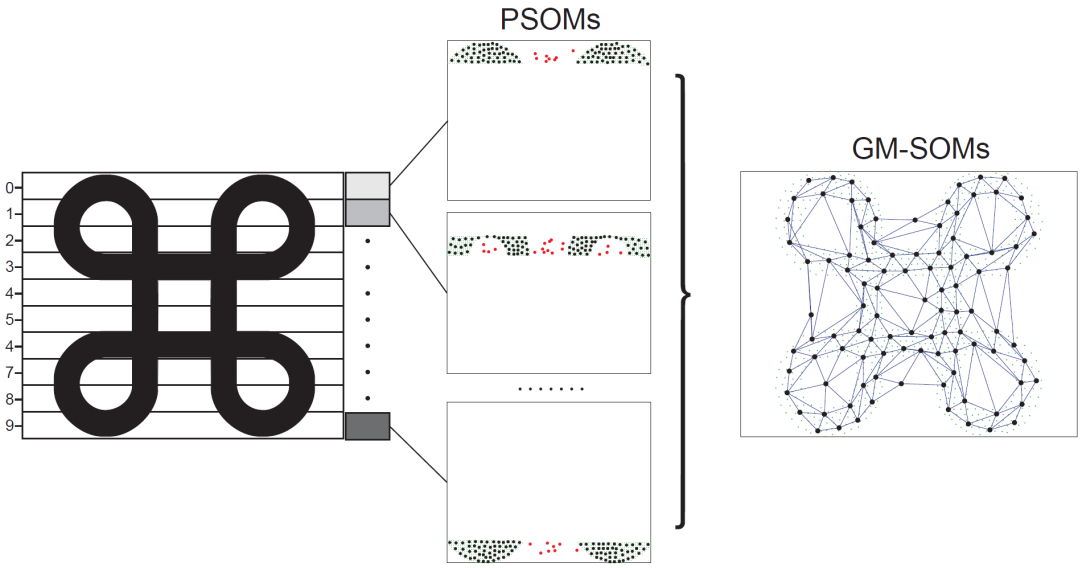
$$k \gg N * p \tag{8}$$

b) Výpočet v jednotlivých uzlech

Na každou část trénovací množiny je aplikován klasický proces učení SOM tak, jak je popsán v kapitole 3.2. Všechny části mají na začátku váhy inicializované na stejné hodnoty a stejnou strukturu sítě. Počet epoch může být vzhledem k počtu epoch použitých při učení pomocí jednoho uzlu nižší, a to až třetinový.

c) Spojení jednotlivých částí

V posledním kroku je vytvořena nová trénovací množina. Tato nová množina je tvořena



Obrázek 9: Znázornění metody GM-SOM [6]

váhovými vektory, které byly vypočteny v rámci každé části. Váhový vektor tvoří hodnoty vah mezi BMU a vstupní vrstvou. Použijí se pouze ty váhové vektory, které odpovídají neuronům alespoň jednou zvolených jako BMU.

Nově vytvořená trénovací množina se použije pro naučení nového modelu SOM se stejnými parametry a takto naučená síť již představuje výsledek celé operace.

Velkou výhodou tohoto přístupu je naprostá vzájemná nezávislost jednotlivých výpočetních uzlů v průběhu výpočtu. Na každém uzlu může být také zvoleno jiné nastavení učících parametrů, což vnáší do výpočtu velkou variabilitu a může přinést zajímavé výsledky.

Na obrázku 9 lze vidět ilustrace paralelizace pomocí GM-SOM. Vstupní množina je v tomto případě rozdělena do deseti částí. Každá z těchto částí je zpracována na samostatném uzlu [6].

4.4.1 Ideální počet uzlů pro GM-SOM

Jak již bylo dříve naznačeno, při využití algoritmu GM-SOM je velmi důležité zvolit vhodný počet uzlů, tak abychom velikostí trénovací množiny v posledním kroku zbytečně nedegradovali úsporu získanou rozprostřením mezi více výpočetních uzlů. Tuto skutečnost lze nejlépe ilustrovat na příkladu.

Mějme trénovací množinu složenou z 60 000 vstupů. Pro naučení chceme použít SOM, která bude mít výstupní vrstvu ve tvaru mřížky 30 krát 30 neuronů. Celkově tedy máme v naší síti 900 neuronů. Pro jednoduchost vezmeme, že 5% neuronů není nikdy zvolených jako BMU, z každého výpočetního uzlu je tedy do trénovací množiny v posledním kroku vygenerováno 855 položek.

V tabulkách 2 a 3 je vidět velikost dávky, která je použita pro každý uzel a kolik položek obsahuje nová trénovací množina. Poslední řádek tabulek ukazuje počet trénovacích vstupů,

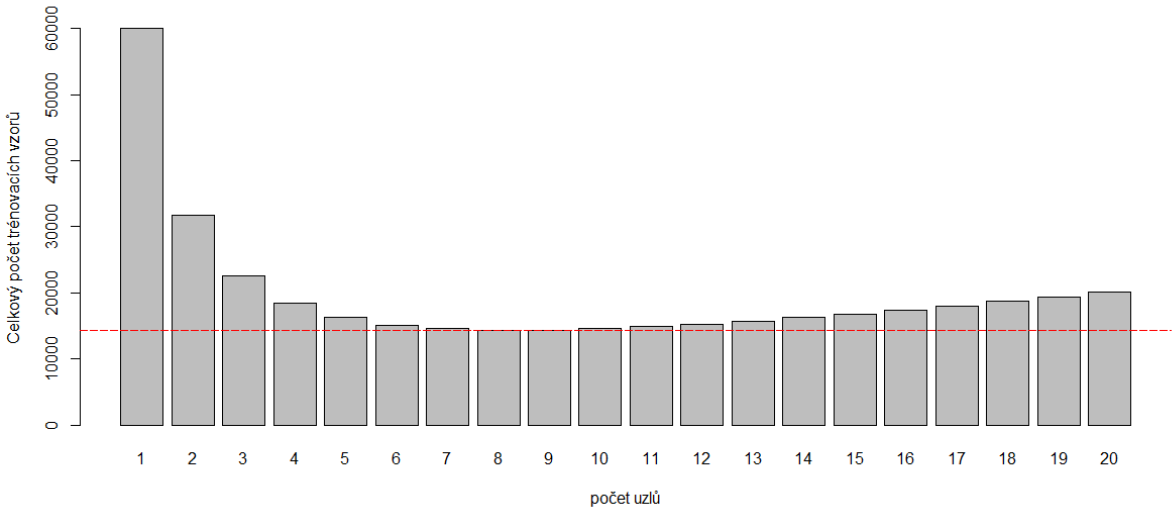
Tabulka 2: Velikosti trénovací množiny dle počtu uzlů, počet uzlů 1 - 10

Počet uzlů	1	2	3	4	5	6	7	8	9	10
Dávka	60 000	30 000	20 000	15 000	12 000	10 000	8 572	7 500	6 667	6 000
Nová TM	x	1 710	2 565	3 420	4 275	5 130	5 985	6 840	7 695	8 550
Celkem	60 000	31 710	22 565	18 420	16 275	15 130	14 557	14 340	14 362	14 550

Tabulka 3: Velikosti trénovací množiny dle počtu uzlů, počet uzlů 11 - 20

Počet uzlů	11	12	13	14	15	16	17	18	19	20
Dávka	5 455	5 000	4 616	4 286	4 000	3 750	3 530	3 334	3 158	3 000
Nová TM	9 405	10 260	11 115	11 970	12 825	13 680	14 535	15 390	16 245	17 100
Celkem	14 860	15 260	15 731	16 256	16 825	17 430	18 065	18 724	19 403	20 100

kterým by měla odpovídat celková délka učení. Tento počet je součet počtu vstupů v dávce a počet vstupů v nově vytvořené trénovací množině. Počítáme, že všechny uzly stráví učením dávky stejný čas.



Obrázek 10: Velikost celkového počtu trénovacích vzorů v závislosti na počtu uzlů

Z grafu na obrázku 10, který je vygenerován na základě dat v tabulce 2 a 3, je vidět, že v tomto konkrétním případě by nejvhodnějším počtem uzlů bylo 8. Do tohoto počtu uzlů stále dochází ke snižování celkového počtu vzorů a pro 9 a více výpočetních uzlů již vidíme nárůst v celkové velikosti trénovacích vzorů, čímž dochází k prodlužování času potřebného k učení. Nejnižší hodnota pro 8 uzlů je v grafu znázorněna také červenou linií.

5 Datové sady

Datové sady, označované také jako datasety, jsou kolekce dat a můžou být určené pro zpracování pomocí neuronových sítí. Datové sady se skládají z jednotlivých objektů, nazývaných jako vzory, pro které jsou charakteristické jednotlivé atributy získané objektivním měřením. Známé datové sady také hrají důležitou úlohu při modifikacích již stávajících algoritmů NS, protože je na nich dobře zmapované chování těchto NS a lze jej považovat za referenční [8].

Nyní budou popsány datové sady použité pro experimenty v rámci této diplomové práce.

5.1 Fisherův dataset kosatců

Tato datová sada patří mezi jedny z nejstarších používaných v oblasti strojového učení, v anglické literatuře lze tuto datovou sadu nalézt také pod názvem *Iris dataset*. Jedná se o sbírku rozměrů částí rostlin z rodu kosatců. Každý záznam odpovídá jiné rostlině. Změřeny byly tyto rozměry: délka a šířka kališních a okvětních lístků. Celkem obsahuje datová sada 150 změřených vzorků, které odpovídají třem různým druhům kosatce. Pro každý druh je zde 50 vzorků [8].

Tabulka 4 shrnuje nejdůležitější atributy této datové sady. Datová sada je dostupná ke stažení na [19]. Tato datová sada neobsahuje rozdělení na testovací a trénovací množinu Toto rozdělení bylo uděláno dodatečně a to na 100 vzorů v trénovací množině a 50 vzorů v testovací množině.

Tabulka 4: Charakteristika datové sady kosatců

Počet vzorů	Počet atributů	Typ atributů	Počet kategorií
150	4	Reálná čísla	3

5.2 Ručně psané číslice

Další z použitých datových sad využitých k pozdějším experimentům je datová sada obsahující vzorky ručně psaných číslic. V datové sadě se nacházejí vzorky od celkem 43 lidí, kde vzorky od 30 lidí tvoří tréninkovou sadu dat a vzorky od zbylých 13 tvoří testovací množinu.

Tato datová sada vznikla na základě sady poskytované společností NIST. Původní data se skládaly z číslic zaznamenaných v rozlišení 32x32 pixelů. Skupiny těchto pixelů byly rozděleny do nepřekrývajících se bloků o rozměru 4x4 pixelů, následně byly pro každý tento blok určen počet aktivních pixelů. Takto vznikla matice 8x8, kdy každý element může nabývat hodnot 0 až 16. Tímto způsobem došlo k zredukování dimenze problému z 1024 vstupních atributů na 64 [16].

Tabulka 5: Charakteristika datové sady psaných číslic

Počet vzorů	Počet atributů	Typ atributů	Počet kategorií
5620	64	Celá čísla, $\langle 0; 16\rangle \subset \mathbb{N}$	10

Charakteristika datové sady je shrnuta v tabulce 5. Datová sada obsahuje celkem 5620 vzorů rozdělených na 3823 v trénovací množině a 1797 v testovací množině. Každý vzor představuje jednu z deseti kategorií, které odpovídají číslicím 0 až 9. Datovou sadu lze stáhnout z [16].

5.3 MNIST

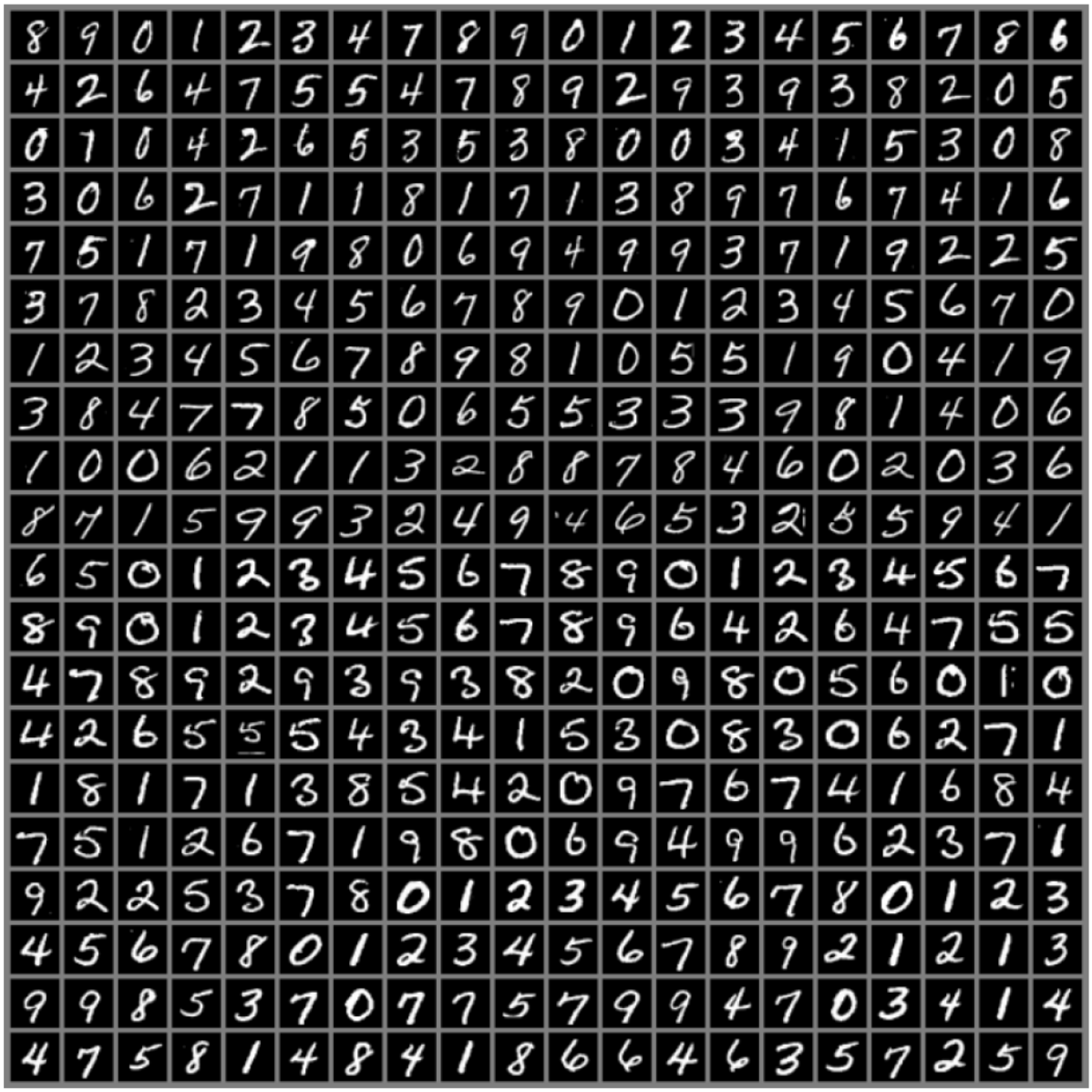
MNIST je datová sada také obsahující vzorky ručně psaných číslic, je však mnohem rozsáhlejší než předchozí případ. Data také vycházejí z dat společnosti NIST, která tyto vzorky sesbírala. Písmeno M v názvu této datové sady odkazuje na anglické *modified*. Datová sada tedy byla předzpracována, došlo k vycentrování a normalizaci velikosti, pro jednodušší použití [8].

Základem této datové sady jsou vzory vycházející z naskenovaných ručně psaných číslic, kde každý vzor obsahuje informaci, které číslici odpovídá. Stručná charakteristika je v tabulce 6. Datová sada MNIST se skládá z 70000 vzorů, kde 60000 obsahuje trénovací množina a 10000 obsahuje testovací množina. Každý vzor se skládá ze 784 atributů a označení kategorie vzoru. Datová sada je dostupná stránce [15].

Tabulka 6: Charakteristika MNIST

Počet vzorů	Počet atributů	Typ atributů	Počet kategorií
70000	784	Celá čísla, $\langle 0; 255 \rangle \subset \mathbb{N}$	10

Na obrázku 11 jsou znázorněny příklady vstupů pro datovou sadu MNIST.



Obrázek 11: Příklad vstupních vzorů datové sady MNIST

6 Modeler neuronových sítí

Jedním z cílů této diplomové práce je rozšířit program Modeler neuronových sítí. Jedná se o software pro podporu výuky předmětu Neuronové sítě, který je vyučován na Vysoké škole báňské - Technické univerzitě v Ostravě. Tento software je zaměřený na vizualizaci vnitřní struktury sítí, zobrazení průběhu učení a sledování odezvy sítě, tak aby bylo pro studenty co nejjednodušší pochopit způsob učení a chování jednotlivých typů NS.

Program Modeler neuronových sítí je implementován v programovacím jazyce JAVA. Jelikož se jednalo již o existující software, bylo první nutné zanalyzovat strukturu kódu tak, aby bylo možné zakomponovat nový modul pro Kohonenovy mapy. Ve výchozím stavu obsahoval MNS pouze modul pro vícevrstvé sítě s učícím algoritmem backpropagation.

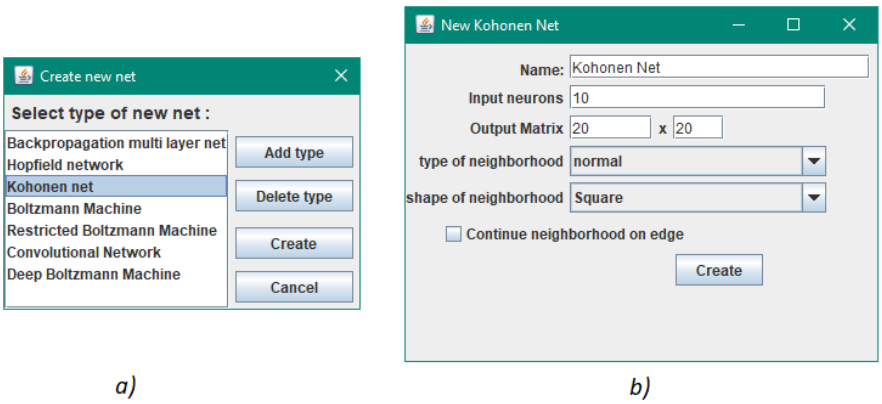
Během vývoje bylo používáno vývojové prostředí Eclipse[11] společně se systémem pro správu verzí GIT[7].

6.1 Popis ovládání a grafického rozhraní

Po spuštění programu se zobrazí úvodní obrazovka, kterou můžeme rozdělit na tři hlavní části. První část představuje menu programu. Pomocí toho menu lze vytvořit novou, či načíst již uloženou neuronovou síť. Pokud uživatel tímto způsobem přidá do programu NS, přichází na řadu další část, ve které jsou zobrazeny aktuální NS dle jejich názvu. V poslední části se zobrazuje detail pro jednotlivé uživatelské akce.

6.1.1 Vytvoření, uložení a načtení již existující SOM

Pokud chce uživatel vytvořit novou síť typu SOM, musí v menu zvolit položku File a poté New. Objeví se okno s nabídkou všech NS implementovaných v rámci programu MNS. Tato nabídka je zobrazena na obrázku 12a. Ze seznamu zvolí uživatel Kohonenovu mapu a pokračuje tlačítkem *Create*.



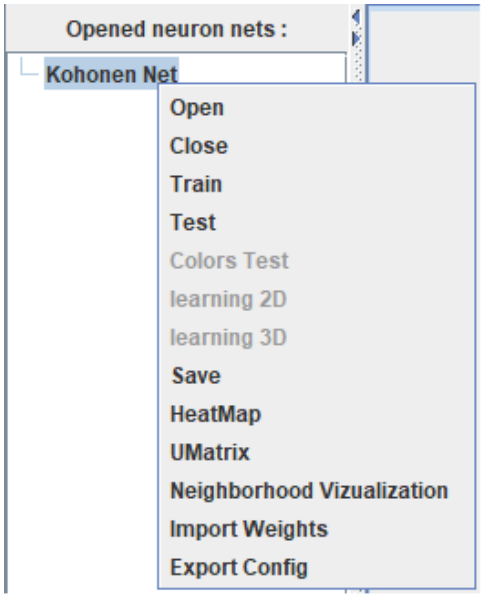
Obrázek 12: Vytvoření nové NS typu SOM v MNS

Další obrazovka umožní uživateli nastavit jednotlivé vlastnosti a strukturu SOM. Obrazovku znázorňuje obrázek 12b. Uživatel má možnost nastavit strukturu sítě v podobě počtu neuronů ve vstupní a výstupní vrstvě. Neurony výstupní vrstvy jsou zadávány v podobě velikosti dvourozměrné mřížky. Tuto strukturu sítě již později není možné změnit.

Dále lze zvolit typ a tvar sousedství, které je následně využíváno při učení. Poslední volba se také týká sousedství a určuje, jestli se výběr sousedů zastaví na konci mřížky výstupních neuronů nebo ne, jak ukazuje obrázek 6. Pro vytvoření pak stačí stisknout tlačítko *Create*.

Takto vytvořená síť se přidá do seznamu sítí v levé části hlavního okna programu MNS a je pro ní dostupná kontextová nabídka, kterou lze vyvolat pravým kliknutím myši na název dané sítě. Jednotlivé možnosti v kontextové nabídce zobrazuje obrázek 13.

Jednou z možností v této nabídce je také uložení, díky tomu lze vytvořenou a například i naučenou síť uložit. Síť se uloží do souboru s koncovkou net. Takto uloženou síť lze později znova do programu MNS načíst a to přes nabídku *File* a *Open*.



Obrázek 13: Nabídka dostupná pro jednotlivé sítě

6.1.2 Změna vlastností SOM

Parametry týkající se sousedství, které jsou nastaveny během vytváření nové sítě, lze později změnit v detailech sítě. Panel s detaily sítě může uživatel vyvolat z kontextové nabídky možností *Open*. V tomto panelu se zobrazí vlastnosti sítě jako počet vstupních a výstupních neuronů, název sítě a všechny volby týkající se sousedství. K zavření tohoto panelu slouží možnost *Close*.

6.1.3 Učení a testování

Další položka kontextového menu vyvolá dialog sloužící k naučení aktuální sítě. Pro potřeby učení zde může uživatel nastavit tři parametry. Prvním je poloměr sousedství, označen R, dále počáteční hodnotu učícího parametru, označen U. Jako poslední lze nastavit parametr označený jako $alfa\,U$, sloužící pro aktualizaci předchozích dvou parametrů během procesu učení.

Obrazovku učení můžeme rozdělit na dvě části, dle typu vstupních souborů s trénovací a testovací množinou. První část je pro vstupní soubory ve formátu, kdy každému vstupu odpovídá jeden řádek a hodnoty jsou vzájemně odděleny tabulátorem či mezerou. Jednotlivé vstupní hodnoty musí být reprezentovány celým nebo desetinným číslem. Pokud obsahuje trénovací množina také kategorii či popis daného vstupu, musí být poslední hodnotou na řádku. Kategorie musí být řetězec znaků a číslic, a zároveň nesmí obsahovat mezery. Pro tyto typy souborů lze zvlášť vybrat soubor s trénovací množinou a zvlášť soubor představující testovací množinu. Pokud není kategorie vstupu součástí trénovací množiny, je možné rozdělení kategorií nahrát dodatečně. I v tomto případě musí mít stejnou strukturu v podobě vstupních hodnot a poslední hodnota určuje kategorii, které vstup odpovídá.

Druhá část obrazovky je určena speciálně pro učení dat z datasetu MNIST. Tento dataset je detailně popsán v kapitole zabývající se jednotlivými datasety. Jelikož se tento dataset skládá z více souborů, musí uživatel zvolit složku, kde se tyto soubory nachází. Je také možnost tento dataset exportovat do souboru, který odpovídá formátu popsanému v předchozím odstavci.

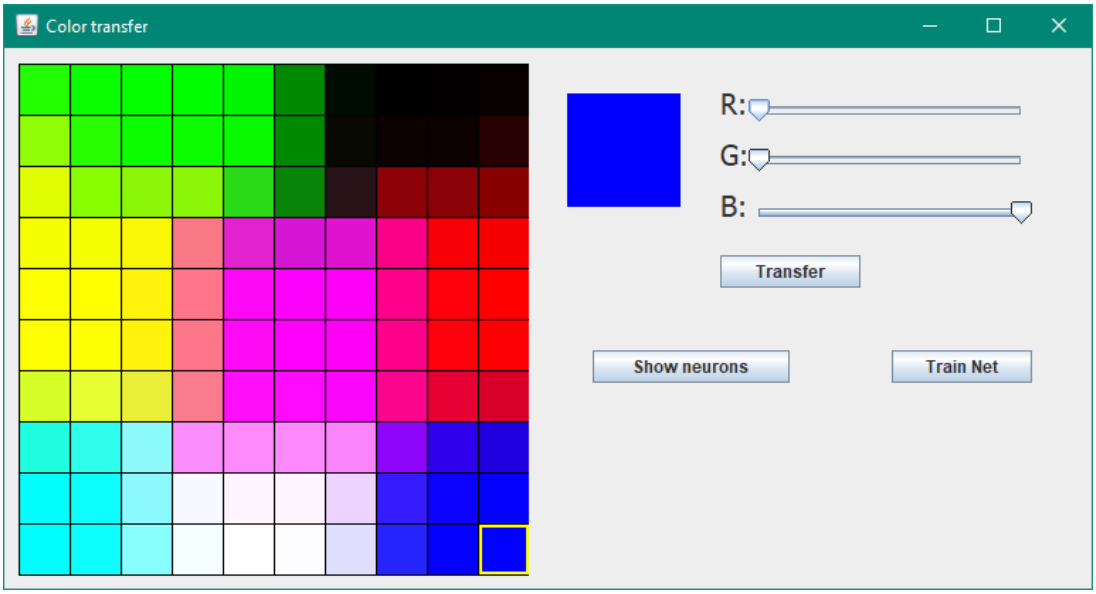
Síť naučenou pomocí jednoho ze dvou výše zmíněných postupů lze také otestovat. Jednou z možností je manuální předložení jednoho vzoru, pomocí dialogového okna dostupného z kontextové nabídky pod názvem test. Toto okno se skládá ze sloupce přestavujícího axony, neboli jednotlivé vstupy. U jednotlivých vstupů může uživatel nastavit hodnotu. Pro takto nastavený vstup lze vynutit odezvu sítě znázorněnou v podobě excitovaného neuronu. Excitovaný neuron se zobrazí v seznamu neuronů výstupní vrstvy s hodnotou 1.

Pro rozsáhlejší testování lze použít test nacházející se v okně pro učení. Zde je možné vybrat soubor s testovací množinou a dojde k vyhodnocení odezvy sítě pro všechny testovací vstupy. Výsledek je pak znázorněn pomocí okna zobrazujícího celkovou úspěšnost odpovědi sítě a detailní rozpis úspěšnosti pro jednotlivé kategorie vstupů.

6.1.4 Vizualizace průběhu učení a odezvy sítě

Jak již bylo dříve zmíněno, nástroj MNS je určen převážně pro podporu výuky NS. Za tímto účelem byla pro sítě typu SOM implementována sada nástrojů poskytující pohled na průběh učení a odezvy sítě.

První z těchto nástrojů je test odezvy, který umožní uživateli vyzkoušet jak se mění excitovaný neuron ve výstupní vrstvě v závislosti na předloženém vstupu. Vstupem sítě jsou tři složky barevného modelu RGB, proto je tato možnost dostupná pouze pro sítě se třemi neurony ve vstupní vrstvě. Každý vstupní neuron odpovídá jedné ze tří základních barev - červené, zelené



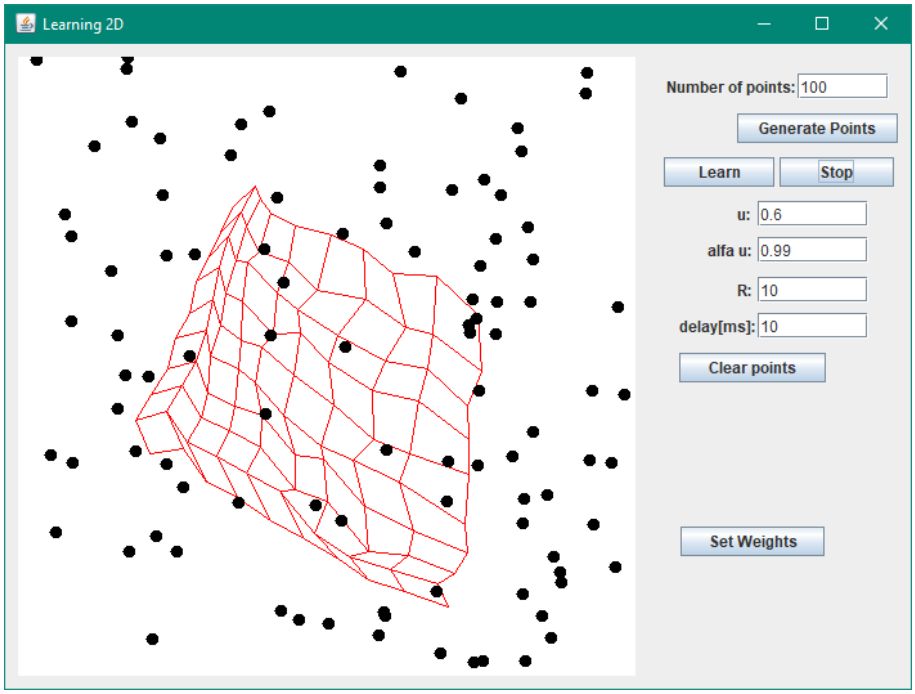
Obrázek 14: Obrazovka pro test odezvy sítě s excitovaným neuronem

a modré. Pokud jako trénovací množinu použijeme 8 vstupních vektorů odpovídajících těmto barvám: černé, červené, zelené, modré, žluté, purpurové, azurové a bílé, může vizualizace výstupní mřížky vypadat podobně jako na obrázku 14. Barvy jednotlivých neuronů odpovídají vahám mezi daným neuronem a neurony výstupní vrstvy. V pravé části obrázku 14 lze vidět tři posuvníky, každý odpovídající jedné ze složek modelu RGB, pomocí kterých může uživatel namíchat vlastní barvu a použít ji jako vstup. Poté dojde k aktivaci sítě a odezva se zobrazí v podobě zvýrazněného excitovaného neuronu.

Další dva nástroje umožňují náhled na chování sítě v průběhu učení. V obou nástrojích je vždy vykreslena výstupní vrstva neuronů pomocí červených přímek. Těmito přímkami jsou vždy spojeny dva sousední neurony. Trénovací množina je zobrazena pomocí bodů na ploše, respektive v prostoru. Položka pro zobrazení učení ve 2D prostoru je dostupná pouze pro sítě se dvěma vstupními neurony a podobně je zobrazení ve 3D prostoru přístupné pouze pro sítě se třemi vstupními neurony.

Obrázek 15 ukazuje vzhled okna pro zobrazení učení ve 2D prostoru. Uživatel zde může manuálně zvolit trénovací množinu bodů pomocí přidání bodů na bílou plochu. Body lze také náhodně vygenerovat. Dále lze nastavit požadované parametry jako učící faktor, velikost sousedství, koeficient snižování učícího faktoru a také prodlevu mezi vykreslením jednotlivých kroků.

Postup pro zobrazení učení ve 3D prostoru se liší tím, že je nutné veškeré parametry nastavit v samostatném okně, a až potom je zobrazeno chování sítě v průběhu učení. Jako trénovací množina je zde použito 10 náhodně vygenerovaných bodů. Během učení lze měnit dobu prodlevy mezi jednotlivými kroky pomocí kláves + a - na klávesnici.



Obrázek 15: Okno zobrazující průběh učení ve 2D prostoru

6.1.5 Zobrazení vnitřního stavu sítě

Po naučení sítě je důležitým prvkem také vizualizace vnitřního stavu, který může pomoci s interpretací výsledků a s odhadem úrovně naučení. Jednou z velmi populárních možností je sjednocená matice vzdáleností, označovaná jako u-matrix. U-matrix zobrazuje vztahy mezi sousedními neurony. Na základě této matice lze graficky reprezentovat vnitřní stav sítě pomocí vzdáleností mezi sousedními neurony. Podle těchto vzdáleností je možné odhalit shluky neuronů.

Hodnota u_{ij} pro jeden neuron se vypočítá dle rovnice 9, kde $u_{ij|i(j-1)}$ je vzdálenost mezi neurony J_{ij} a $J_{i(j-1)}$. Součet vzdáleností se všemi sousedy je vydělen počtem sousedů, v tomto případě jsou sousední neurony 4.

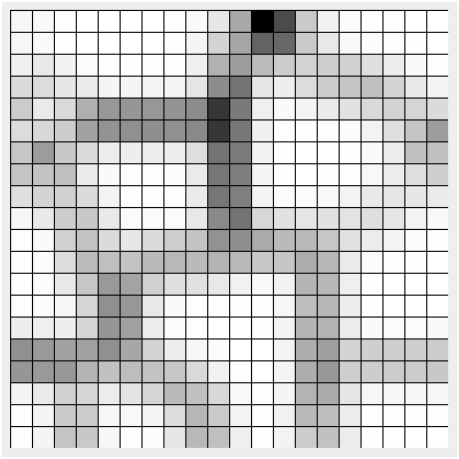
$$u_{ij} = (u_{ij|i(j-1)} + u_{ij|i(j+1)} + u_{ij|(i-1)j} + u_{ij|(i+1)j})/4 \tag{9}$$

Hodnoty pro jednotlivé neurony jsou následně reprezentovány pomocí mřížky v odstínech šedi, kde tmavá barva značí velkou vzdálenost od sousedních neuronů. Světlá barva naopak představuje, že neurony odpovídají vstupům, které jsou ve vstupním prostoru blízko sebe. Na takto zobrazené výstupní vrstvě hledáme světlé oblasti, které představují shluky neuronů, zatímco tmavé oblasti tyto shluky oddělují [22].

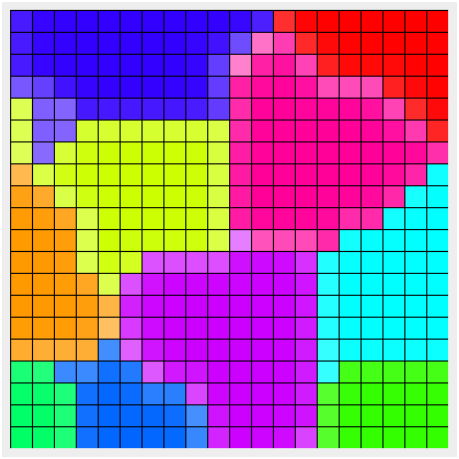
Druhou možností jak reprezentovat shluky neuronů vytvořené při učení je pomocí tzv. teplotní mapy, kde jsou neurony neurony jedné kategorie znázorněny stejnou barvou. V rámci programu MNS je možné buď zobrazit teplotní mapu na základě kategorií, které byly neuronům přiřazeny při učení, nebo pomocí souboru s testovací množinou vstupů, kde jeden vstup

odpovídá jedné kategorii. Pro vytvoření teplotní mapy na základě souboru lze zohlednit také vzdálenosti od vstupních dat. Neurony jedné kategorie s nejbližší vzdáleností od testovacího vstupu jsou obarveny sytějším odstínem, zatímco vzdálenější neurony dané kategorie jsou odstínem světlejším.

Na obrázku 16 je znázorněn příklad sjednocené matice vzdáleností zobrazený v programu MNS. Obrázek 17 zobrazuje teplotní mapu. Oba obrázky jsou vygenerovány pro síť s výstupní vrstvou o rozměrech 20 krát 20 neuronů. Trénovací množina zde představovala čísla 0 až 9 na mřížce 8 krát 8 pixelů. Z obou obrázků jsou jasně viditelné shluky, kde neurony patřící do jednoho shluku představují jednu z 10 číslic.



Obrázek 16: U-matrix



Obrázek 17: Teplotní mapa

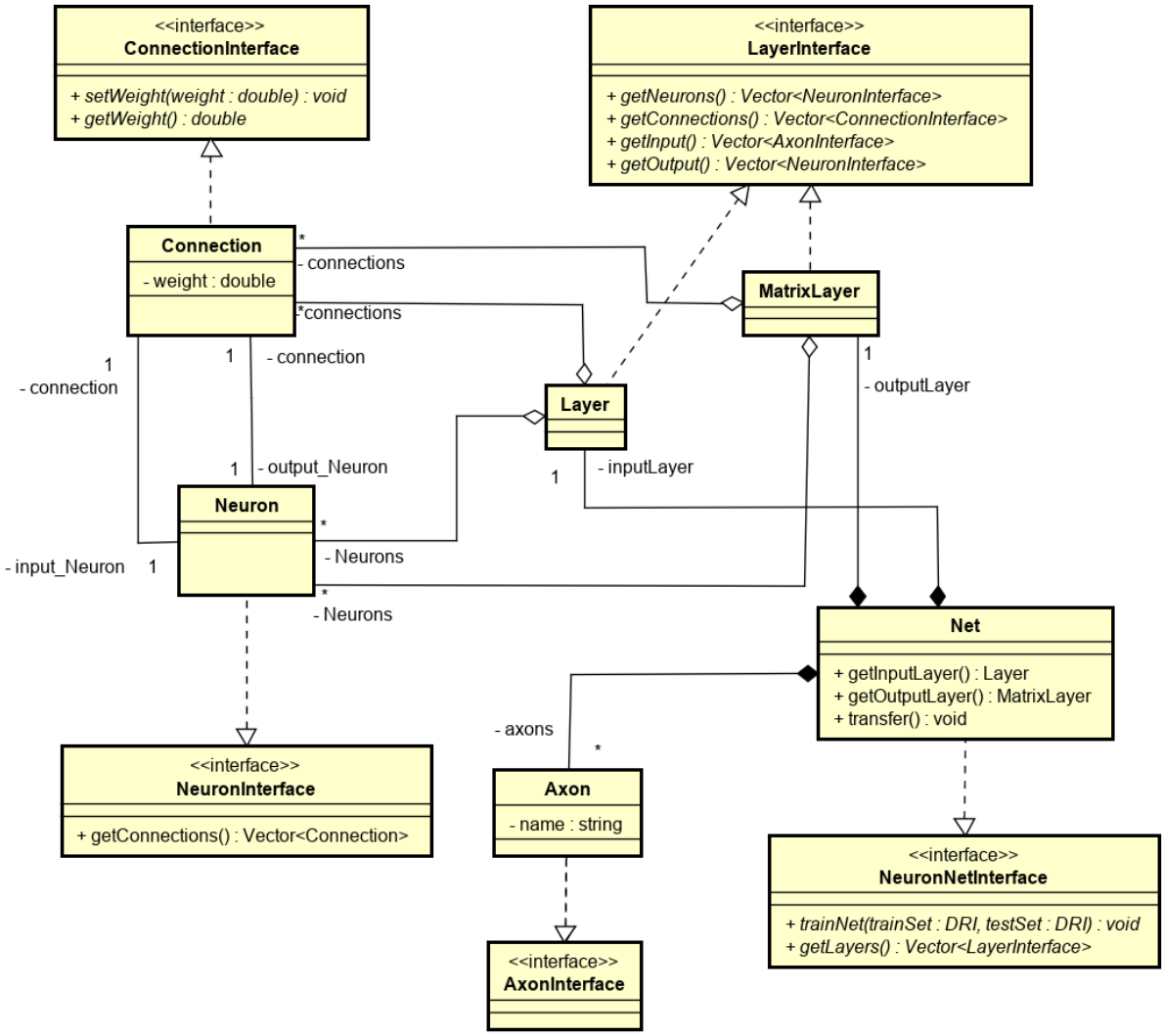
6.1.6 Zobrazení sousedství

V rámci SOM hraje velmi důležitou sousedství, které se uplatňuje během učení sítě. Z tohoto důvodu obsahuje program MNS také možnost zobrazit chování funkce sousedství pro jednotlivé implementované typy.

6.1.7 Propojení s programem pro paralelní učení SOM

Poslední dvě funkce slouží k zajištění kompatibility programu MNS s programem pro paralelní implementaci SOM. První funkcí je importování stavu sítě ze souboru vygenerované z již zmiňovaného programu. Tato funkce umožní exportovanou síť načíst a dále s ní pracovat v rámci programu MNS.

Druhou funkcí je vygenerování konfiguračního souboru obsahující řídící parametry a informace o struktuře sítě pro program ParallelSOM.



Obrázek 18: Třídní diagram po implementaci modulu SOM

6.2 Detaily implementace

Pro zakomponování modulu SOM do stávajícího stavu programu MNS poskytoval rozhraní, která bylo nutné implementovat. Strukturu části programu po zakomponování modelu pro SOM lze vidět na třídním diagramu na obrázku 18. Rozhraní na tomto obrázku již byly v programu přítomné a bylo na ně navázáno. Některé metody a atributy tříd byly v diagramu zanedbány pro zachování přehlednosti, ponechány byly metody a atributy s významnou funkcí v rámci celého programu.

6.2.1 Třída *Net*

Nejdůležitější komponentou modulu je třída *Net*. Tato třída implementuje rozhraní *NeuronNetInterface*. Třída *Net* obstarává vlastní logiku SOM algoritmu, obsahuje metody pro učení

a vyvolání informace.

Třída obsahuje atributy *inputLayer* a *outputLayer*. Atribut *inputLayer* je instancí třídy *Layer* a představuje vrstvu vstupních neuronů. Atribut *outputLayer* je instancí třídy *MatrixLayer* a představuje vrstvu výstupních neuronů uspořádaných do mřížky.

6.2.2 Třídy *Neuron* a *Connection*

Třídy *Neuron* a *Connection* představují základní komponenty neuronové sítě. Třída *Connection* obsahuje atribut představující váhu spoje mezi neurony.

6.2.3 Problém objektového pojetí komponent sítě

Při implementaci SOM modulu, v podobě popsané výše a znázorněné na třídím diagramu na obrázku 18, byly všechny komponenty reprezentovány vlastním objektem. V programu měl každý neuron a spoj mezi neurony vlastní objekt. Tato vlastnost se negativně projevovala na výkonu celého programu, a proto byla experimentálně vyzkoušena alternativní implementace, kdy jsou jednotlivé spoje mezi neurony reprezentovány pomocí trojrozměrného pole desetinných čísel.

Tabulka 7: Doba učení pro implementaci pomocí objektů a polí

Počet neuronů		Vzory	Úspěšnost		Čas [s]	
Vstupních	Výstupních	(trénink/test)	Objekty	Pole	Objekty	Pole
4	10x10	100/50	92,80 %	92,00 %	25,42	1,56
4	20x20	100/50	91,60 %	92,00 %	405,02	5,05
4	30x30	100/50	92,00 %	91,20 %	2015,38	10,46

Porovnání času potřebného k učení sítě lze vidět v tabulce 7. Časy a úspěšnost v tabulce jsou průměrné hodnoty z 5 běhů, k porovnání byl použit Fisherův dataset kosatců. Z tabulky je vidět značné zrychlení učení sítě při změně implementace. Měření bylo prováděno pro síť se 4 vstupními neurony. Rozměr výstupní vrstvy je vidět ve druhém sloupci. Jak ukazuje druhý řádek tabulky, pro množinu skládající se ze 100 trénovacích vzorů se učení, pro síť se 100 neurony ve výstupní vrstvě, zrychlilo více než 16krát. Zrychlení nemělo žádný vliv na úspěšnost rozpoznávání vzorů v testovací množině.

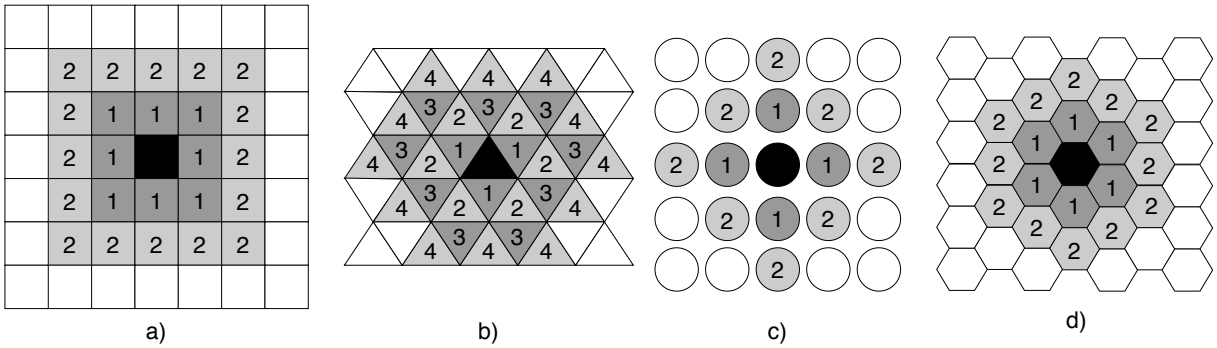
Z důvodu zachování kompatibility modulu s ostatními nástroji programu MNS obsahuje výsledná implementace objektovou reprezentaci komponent společně s reprezentací pomocí pole. Objektová reprezentace struktury sítě se však uplatňuje pouze v nejnutnějších případech a preferována je implementace pomocí pole.

6.2.4 Funkce sousedství

Modul SOM obsahuje celkem čtyři různé typy sousedství, jedná se o:

- čtvercové, obrázek 19a,

- trojúhelníkové, obrázek 19b,
- kruhové, obrázek 19c,
- šestiúhelníkové, obrázek 19d.



Obrázek 19: Typy sousedství pro SOM v programu MNS

Na obrázku 19 jsou znázorněny jednotlivé typy sousedství implementované v programu MNS. Číslice v každém z neuronů označuje poloměr sousedství odpovídající danému neuronu. Sousedé jsou zde vybíráni pro černě zvýrazněný neuron.

Pro každý typ sousedství je v programu možné vybrat také způsob, jakým způsobem jsou neurony spadající do sousedství ovlivněny. První možností je, že jsou všechny neurony přitahovány stejně, v tomto případě odpovídá aktualizace vah rovnici 6. Další dva způsoby zohledňují vzdálenost v jaké se neuron od BMU nachází a podle toho klesá také míra ovlivnění. Čím vzdálenější neuron, tím menší mírou dojde k jeho ovlivnění. Tento způsob si můžeme představit jako přidání dalšího parametru P, jak znázorňuje rovnice 10. Tento parametr může nabývat hodnot od 0 do 1, respektive od 0,5 do 1, což záleží na zvolené variantě. Pokud tedy neuron přímo sousedí s BMU bude mít parametr P hodnotu 1.

$$w_{ij}^{new} = w_{ij}^{old} + \alpha P(x_i - w_{ij}^{old}) \tag{10}$$

6.2.5 Průběh učení ve 3D

Pro zobrazení průběhu učení ve 3D byl použit herní engine jMonkeyEngine [12]. Jedná se o otevřený herní engine, který je určený pro programovací jazyk Java a vývoj 3D her.

7 Program pro paralelní učení SOM

Jako druhý cíl této diplomové práce bylo vytvoření programu pro paralelní průběh učení algoritmu SOM, dále nazýván také jako ParallelSOM. Tento program lze považovat za rozšíření programu MNS představeným v předchozí kapitole.

Z důvodu co nejjednodušší kompatibility s MNS je tento software také implementován v programovacím jazyce JAVA. ParallelSOM neobsahuje grafické uživatelské rozhraní, jedná se pouze o konzolovou aplikaci, jelikož se předpokládá spouštění na serverových výpočetních uzlech, kde nemusí být dostupné grafické prostředí. K vývoji bylo rovněž použito vývojové prostředí Eclipse[11] a systém pro správu verzí GIT[7].

Tato část diplomové práce využívá a rozšiřuje poznatky nastíněné v rámci práce *Rozšiřující knihovna pro modeler neuronových sítí* [17].

7.1 Popis ovládání

Pro nastavení a spuštění programu ParallelSOM hrají důležitou roli spouštěcí argumenty. Program může být spuštěn ve třech režimech, a to v režimu Master, v režimu Slave a v automatickém režimu. Všechny se vzájemně odlišují počtem argumentů, první argument však vždy určuje režim, v jakém bude program spuštěn. Každý režim nyní bude detailněji popsán zvlášť.

7.1.1 Režim Master

V tomto režimu se instance programu chová jako řídící prvek a čeká na připojení dalších instancí v režimu Slave. V rámci učení jedné neuronové sítě je vždy pouze jedna instance programu v režimu Master, zbylé jsou v režimu Slave.

Tento režim řídí veškerou komunikaci a drží aktuální stav NS. Stará se také o distribuci stavu NS ostatním uzlům a o zpracování dat poskytnutých z jednotlivých výpočetních uzlů. Režim Master se také stará o export výsledků po dokončení učení a neprobíhá na něm samostatný výpočet.

Pro spuštění programu v režimu Master je nutné jako první spouštěcí argument uvést písmeno M jako Master. Další argumenty poskytují informace pro samotný průběh učení a musí být dodržené jejich přesné pořadí v jakém jsou zadány. Druhým argumentem je cesta k souboru s konfigurací struktury a parametrů NS. Tento soubor je vygenerován z programu MNS. Dalším v pořadí třetím argumentem je název souboru pro export výsledků. Čtvrtým argumentem musí být cesta k souboru obsahující trénovací množinu. Pátým argumentem je velikost učící dávky a za tímto argumentem následuje typ učení SOM. Poslední sedmý argument je nepovinný a jedná se o specifikaci čísla portu, na kterém bude probíhat komunikace s ostatními uzly.

Jednotlivé typy učení SOM a jejich rozdíly jsou vysvětleny v samostatné části.

7.1.2 Režim Slave

Na rozdíl od režimu Master, který se stará pouze o řízení běhu učení, zajišťuje instance v režimu Slave samotné učení. Instancí programu v režimu Slave může být více, kdy každá část běží paralelně a učení probíhá na základě dat obdržených od instance Master. Je možné také spustit více instancí typu Slave v rámci jednoho programu, kdy každá instance běží v samostatném vlákně. K určení počtu vláken slouží jeden z argumentů.

Argumenty potřebné pro spuštění v režimu Slave jsou následující : typ režimu S jako Slave, počet vláken, IP adresa Master uzlu, cesta k souboru s konfigurací NS, cesta k souboru s trénovací množinou a nepovinný port pro komunikaci.

Aby bylo možné uskutečnit komunikaci mezi instancemi typu Slave a Master, je nutné jako jeden z argumentů při spouštění instance typu Slave zadat IP adresu s portem uzlu, na kterém je spuštěna instance Master. Pokud je zvoleno více vláken a specifikován port, je tento port použit pro první instanci typu Slave a pro další instance je tato hodnota inkrementována.

7.1.3 Automatický režim

Poslední z dostupných režimů je automatický režim. Jedná se o částečnou kombinaci dvou předchozích, kdy je automaticky zvoleno, zda je třeba pustit instanci typu Master nebo Slave. Tento režim nabízí větší uživatelskou přívětivost, jelikož není nutné zjišťovat IP adresu Master uzlu. V tomto režimu dojde k automatickému zjištění, zdali už běží instance typu Master nebo ne. Pokud zatím není spuštěna žádná instance typu Master, dojde k její spuštění. Pokud je již instance typu Master spuštěna, spustí se automaticky instance typu Slave. Obě instance se následně chovají stejným způsobem jako bylo popsáno dříve.

Stejně jako v případě režimu Slave, je i zde možnost specifikovat počet vláken, kdy může instance typu Master běžet v jednom a v dalších budou spuštěny instance typu Slave.

Pro spuštění tohoto režimu je třeba jako první argument zadat písmeno A. Následuje argument počet vláken, cesta k souboru s adresou Master uzlu,k souboru pro export, k souboru s konfigurací SOM, k souboru s trénovací množinou, velikost dávky a typ učení SOM. I zde je možné jako nepovinný argument zvolit číslo portu.

Cesta k souboru pro adresu Master uzlu je hlavní rozdíl od spouštění předchozích dvou režimů. Pomocí tohoto souboru dochází automatickému spouštění Master a Slave uzlů. Pokud zatím není spuštěn uzel v režimu Master, vytvoří soubor do kterého zapíše svou adresu a dojde k jeho spuštění. Následně spuštěné uzly již pouze tuto adresu přečtou a spustí se v režimu Slave.

Tento režim lze také využít k sekvenčnímu učení SOM a to v případě, kdy je jako počet vláken zvoleno pouze jedno. V tomto případě zůstávají stejné argumenty.

7.1.4 Typy učení

Uživatel si může zvolit mezi jedním ze čtyř typů učení, hodnoty argumentu pro jednotlivé typy jsou zobrazeny v tabulce 8.

Prvním typem je GM-SOM. Učení tedy probíhá tak, jak je popsán v kapitole 4.4. Ve zkratce, trénovací množina se rozdělí na části dle velikosti zadané dávky, na každý Slave uzel jsou rozeslány informace ohledně rozsahu vzorů z trénovací množiny a výchozí stav NS. Po naučení daných vzorů jsou zpět na řídící uzel odeslány váhové vektory neuronů. Řídící uzel vytvoří z těchto vektorů novou trénovací množinu, a jakmile obdrží data od všech výpočetních uzlů, zahájí se učení nové trénovací množiny na Master uzlu. Komunikace mezi řídícím a ostatními uzly probíhá pouze na začátku a po skončení zpracování dat. V průběhu učení k žádné komunikaci nedochází.

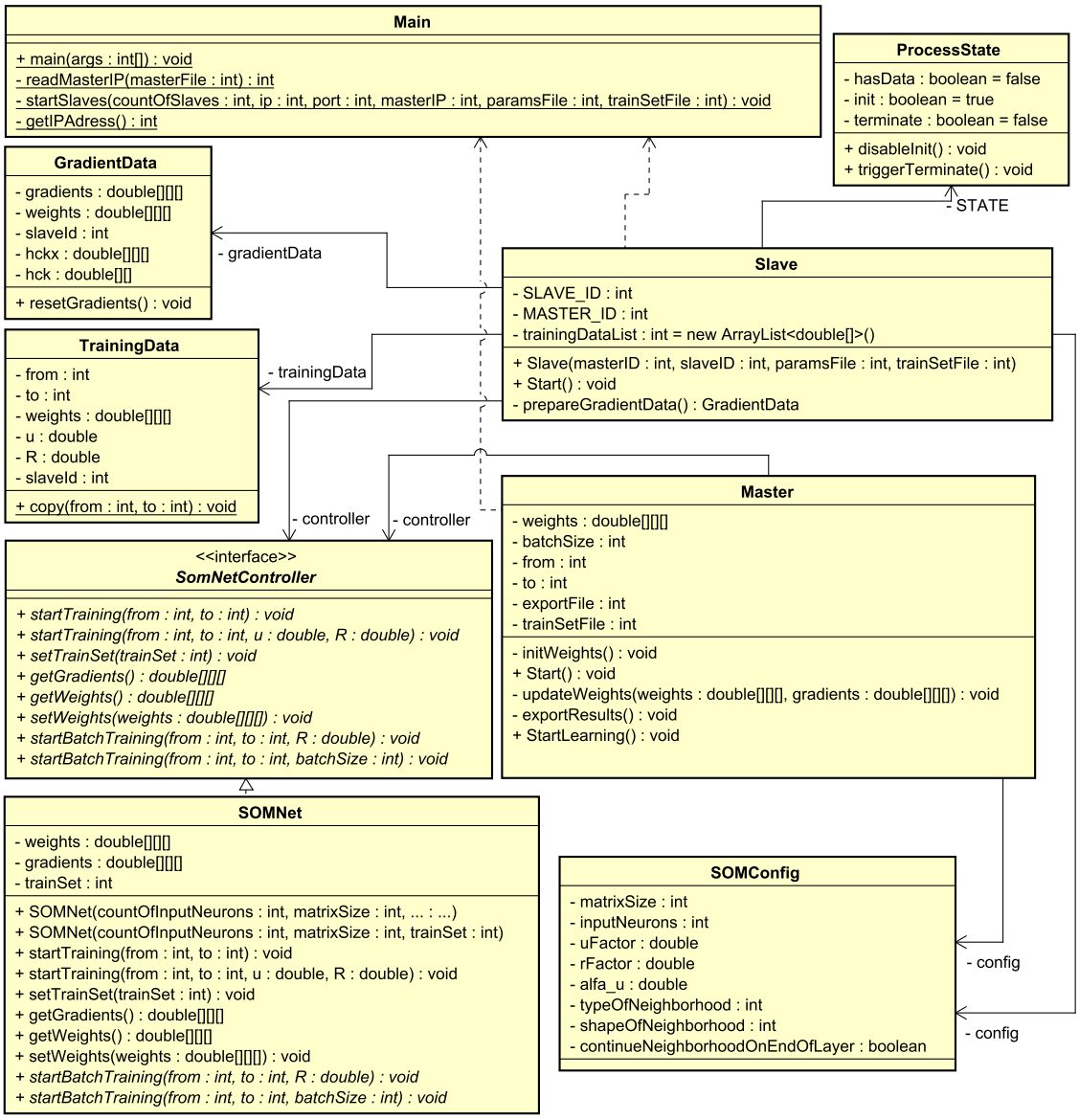
Tabulka 8: Typy učení a odpovídající hodnoty argumentů

Hodnota argumentu	Typ učení
0	GM-SOM
1	Datový paralelismus
2	Datový paralelismus, jedna iterace
3	Dávková SOM

Další dva typy jsou založeny na obecném datovém paralelismu, tak jak se aplikuje například u sítí s učícím algoritmem backpropagation. V prvním případě řídící uzel postupně rozesílá části trénovací množiny na jednotlivé výpočetní uzly. Na každé části proběhne celý algoritmus učení SOM a následně jsou zpět na řídící uzel odeslány změny jednotlivých vah oproti původnímu stavu. Řídící uzel tyto změny aplikuje na svou strukturu NS a jakmile dokončí výpočty všechny a jsou aplikovány veškeré změny dojde k exportu výsledné NS. Každý výpočetní uzel má vlastní řídící parametry. Celá tréninková množina je v tomto případě zpracována právě jednou. Komunikace mezi uzly je zde stejně jako v předchozím případě uskutečněna pouze na začátku a při odeslání dat po konci učení. Tomuto způsobu odpovídá hodnota argumentu při spouštění 1.

Druhým případ je velmi podobný s předchozím, hlavní rozdíl je však v průběhu učení na výpočetních uzlech. Pro zadaný rozsah vzorů z trénovací množiny je provedena pouze jedna iterace v rámci učení. Řídící parametry jsou globálně řízeny Master uzlem. Komunikace mezi uzly zde probíhá po každé provedené iteraci na výpočetním uzlu. Řídící uzel aplikuje změny vah přijaté od výpočetního uzlu a následně odešle nový rozsah trénovací množiny a aktualizované váhy do výpočetního uzlu. Jakmile jsou splněny ukončovací podmínky výpočtu dojde k dokončení výpočtů na výpočetních uzlech, aktualizace vah a exportování výsledku. Tento přístup lze zvolit hodnotou argumentu 2.

Posledním typem učení je učení v dávkách. Na jednotlivé výpočetní uzly jsou rozeslány části trénovací množiny, ty jsou zde zpracovány a pro jednotlivé neurony jsou vypočteny hodnoty odpovídající čitateli a jmenovateli v rovnici 7. Na základě přijatých hodnot od výpočetních uzlů provede řídicí uzel výpočet nových vah na konci každé epochy.



Obrázek 20: Třídní diagram programu ParallelSOM

7.2 Detaily implementace

Na obrázku 20 je znázorněna struktura programu ParallelSOM pomocí třídního diagramu. Třída *Main* obstará spuštění programu v režimu na základě argumentů. V rámci této třídy dochází k zavolání tříd *Master* a *Slave*, které obstarávají funkce dle režimu se stejným jménem popsaných výše.

7.2.1 Třídy *Master* a *Slave*

Třída Master představuje řídící uzel. Tato třída obstarává řízení běhu učení SOM, distribuci pokynů na výpočetní uzly a export výsledků. Ke spuštění slouží metoda *Start*, kdy dojde k inicializaci knihovny Aeron a vyčkávání na připojení výpočetních uzlů. V rámci konstruktoru jsou inicializovány výchozí váhy NS. V této třídě je také udržována aktuální podoba NS.

Pokud je spuštěno učení pouze na jednom uzlu, tedy bez paralelizace, nevyčkává třída *Master* na připojení dalších uzlů, ale je učení zahájeno rovnou.

Třída *Slave* komunikuje s instancí třídy *Master* pomocí knihovny Aeron, která je popsána v samostatném bodě. Třída *Slave* představuje výpočetní uzel. Každá instance udržuje svou lokální NS, která je vždy před začátkem výpočtu inicializována váhami přijatých od Master uzlu. Přijatá zpráva obsahuje instanci třídy *TrainingData*, která v sobě nese informace potřebné k průběhu učení. Po přijetí dat proběhne cyklus učení a je připravená instance třídy *GradientData*, která je odeslána Master uzlu ke zpracování.

7.2.2 Třídy *TrainingData* a *GradientData*

Tyto třídy slouží k zapouzdření dat, která jsou vzájemně posílána mezi Master a Slave uzlem. Třídy *TrainingData* slouží k zaslání dat potřebných pro učení na Slave uzlu. Obsahuje informace ohledně rozsahu vzorů z trénovací množiny, které má daný uzel zpracovat, dále pak váhy a řídící parametry. Pro kontrolu obsahují také identifikátor Slave uzlu, pro případ, že by došlo k odeslání špatných dat. Mezi uzly nedochází k zasílání dat z trénovací množiny. Jsou zasílány pouze indexy, které identifikují rozsah vzorů z trénovací množiny, aby se omezil objem zasílaných dat.

Třída *GradientData* představuje odeslané výsledky ze Slave uzlu na Master uzel. V závislosti na typu učení obsahuje váhové vektory, v případě GM-SOM, či pouze změny vah, v případě dalších typů. Opět je zde také identifikátor Slave uzlu.

7.2.3 Třída *SOMNet*

Třída SOMNet představuje samotnou implementaci NS typu SOM. Zajišťuje veškeré operace prováděné s NS. Pomocí konstruktoru mohou být nastaveny jednotlivé řídicí parametry. Metoda *startTraining* spouští průběh učení. Metoda *setWeights* slouží k nastavení vah sítě. Metody *getGradients* a *getWeights* poskytují přístup k vahám a jejich změnám, tak aby mohly být dále zpracovány.

7.2.4 Knihovna Aeron

Pro fungování distribuovaného výpočtu bylo třeba zajistit spolehlivou komunikaci mezi jednotlivými částmi. Pro tento účel byla využita knihovna Aeron. Aeron je knihovna poskytující efektivní přenos zpráv dostupná pro Java a C++ aplikace. Je navržena pro práci nad nespolehlivými přenosovými protokoly jako UDP či Infiniband. Návrh knihovny byl zaměřen převážně

na komunikaci s nízkým zpožděním a s co největší propustností. Knihovna Aeron integruje také jednoduché binární kódování pro zvýšení výkonu při kódování a dekódování zasílaných zpráv.

Koncept knihovny spočívá v jednosměrných kanálech pro zasílání zpráv, označovaných jako *Publication* a *Subscription*. *Publication* je kanál do kterého je možné zapisovat, využívá se tedy pro odesílání zpráv, zatímco kanál *Subscription* slouží ke čtení, tedy přijímání zpráv.

Fungování knihovny Aeron nad protokolem UDP může být ve třech módech. První je Unicast, druhým je Multicast a posledním z nich je Multi-Destination-Cast. V rámci této práce je využíván mód Unicast a Multi-Destination-Cast, proto budou rozepsány.

V Unicast módu je uskutečněna komunikace bod-bod. Příjemce poslouchá na specifikované IP adrese a UDP portu. Na tuto specifikovanou IP adresu a port odesílatel zašle zprávu pomocí kanálu *Subscription*. Příjemce následně odešle zprávy o stavu a zprávy o negativním potvrzení zpět odesílateli. Tento přístup je využit při komunikaci Slave uzlu s Master uzlem, kde dochází k zasílání výsledků zpět řídícímu uzlu.

Multi-Destination-Cast kombinuje režimy Unicast společně s Multicast. Odesílatel spravuje skupinu příjemců jako multicast skupinu, avšak pro adresování a odesílání zpráv je využit Unicast mód. Seznam příjemců může být v tomto případě spravován manuálně a lze příjemce přidávat i odebírat. Režim Multi-Destination-Cast je využit v případě Master uzlu, který spravuje skupinu Slave uzlů a rozesílá jim potřebná data [1].

8 Analýza a testování

Tato kapitola je zaměřena na otestování programu pro paralelní běh SOM v serverovém prostředí a vzájemné porovnání jednotlivých způsobů paralelizace Kohonenových map. Porovnáno bude chování implementovaných způsobů paralelizace vzhledem k počtu výpočetních uzlů a také dle velikosti učící dávky distribuované těmto uzlům. Dále dojde k porovnání jednotlivých typů sousedství a jejich vlivu na úspěšnost učení sítě. Veškerá měření v této kapitole jsou prováděna na superpočítači Salomon.

Tabulka 9: Naměřené hodnoty pro učení bez paralelizace

Algoritmus	Datová sada	Čas [s]	Úspěšnost [%]	Správně	Špatně
On-line SOM	Ručně psané číslice	113	96,55	1735	62
BSOM	Ručně psané číslice	51	88,98	1599	198
On-line SOM	MNIST	5994	92,90	9290	710
BSOM	MNIST	2083	79,30	7930	2070

V tabulce 9 jsou pro porovnání uvedeny naměřené hodnoty pro učení SOM bez využití paralelizace. Z tabulky je vidět, že dávkové učení je rychlejší oproti on-line algoritmu díky méně častější aktualizaci jednotlivých vah, avšak došlo k poklesu úspěšnosti. Pro BSOM byla v obou případech zvolena velikost učící dávky 1000.

Fisherova datová sada kosatců byla využita převážně při vývoji a vzhledem k malému rozsahu dat není vhodná pro testování paralelizace, proto je v této kapitole použita pouze datová sada ručně psaných číslic a dataset MNIST. V tabulce 10 je uvedena struktura sítě a parametry pro jednotlivé datové sady použité při měření. Pro obě datové sady bylo sousedství ukončeno na konci mřížky.

Tabulka 10: Struktura sítě a nastavení parametrů

Datová sada	Počet neuronů		U	R	alfa U	Sousedství	
	Vstupních	Výstupních				tvar	typ
Ručně psané číslice	64	15x15	0,6	7	0,99	čtvercové	$P \in \langle 0; 1 \rangle$
MNIST	784	20x20	0,6	10	0,90	čtvercové	$P \in \langle 0; 1 \rangle$

8.1 Testovací prostředí

Pro otestování paralelizace bylo zvoleno prostředí superpočítače Salomon. Využití programu ParallelSOM je předpokládáno právě na podobných výpočetních jednotkách s velmi vysokým dostupným výkonem. Superpočítač Salomon se skládá z 1008 výpočetních uzlů a z celkem 129 TB RAM. Každý z výpočetních uzlů je osazen 2 výkonnými procesory Intel Xeon, každý s 12 jádry, a 128 GB RAM. Vzájemné spojení mezi výpočetními uzly je zajištěno pomocí InfiniBand sítě.

Tabulka 11: Charakteristika výpočetního uzlu superpočítače Salomon

CPU	2 x Intel Xeon E5-2680v3, 2.5 GHz
Počet jader	2 x 12
RAM	128 GB
OS	CentOS Linux

Operačním systémem je v tomto případě CentOS. Superpočítač Salomon disponuje celkovým výkonem 2011 TFLOP/s. V tabulce 11 jsou shrnuty informace ohledně jednotlivých výpočetních uzlů [4].

8.2 Způsob měření

Superpočítač Salomon umožňuje libovolně konfigurovat počet výpočetních uzlů, na kterých je spuštěna daná úloha. Pro datovou sadu ručně psaných číslic byly pro všechny výpočty využity dva výpočetní uzly, kdy na každém uzlu běžel stejný počet instancí.

Jelikož je datová sada MNIST náročnější na výpočetní čas, bylo v rámci úspor omezeného výpočetního času zvoleno následující schéma výpočtů. Pro výpočty s využitím maximálně 10 instancí byl alokován jeden výpočetní uzel, na kterém běžely jak instance typu Master, tak i všechny instance typu Slave. Pro výpočty, ve kterých bylo využíváno více než 10 instancí, byly alokovány dva výpočetní uzly, stejně jako pro předchozí dataset.

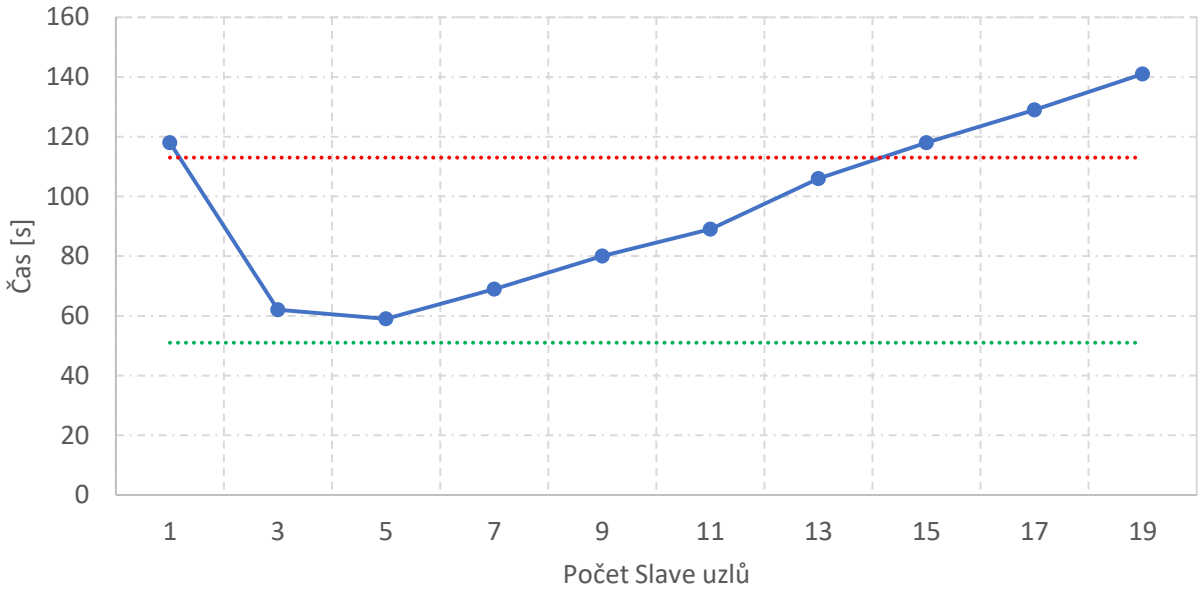
8.3 GM-SOM

Jako první bude provedena analýza algoritmu GM-SOM. Paralelismus u GM-SOM spočívá v rozdělení trénovací množiny mezi jednotlivé výpočetní uzly, proto se od počtu výpočetních uzlů odvíjí i velikost dávky pro každý uzel tak, aby součet dávek pro jednotlivé uzly pokryl celou trénovací množinu.

8.3.1 Datová sada ručně psaných číslic

Tabulka 15 v příloze zobrazuje naměřené hodnoty pro algoritmus GM-SOM a dataset ručně psaných číslic. Z tabulky je patrné, že počet výpočetních instancí nemá v tomto případě vliv na úspěšnost. Úspěšnost se ve všech případech pohybovala v rozmezí mezi 94 a 96 procenty.

Na obrázku 21 je znázorněna doba učení v závislosti na počtu Slave uzlů. Je vidět, že tvar křivky odpovídá předpokládanému vývoji nastíněném v teoretickém příkladu na obrázku 10. V nejrychlejším případě bylo učení dokončeno za 59 vteřin. Ve srovnání s on-line učením se jedná o téměř dvojnásobné zrychlení. Oproti hodnotě času dávkového učení bez paralelizace je však tento čas delší o 8 sekund, ale s vyšší úspěšností rozpoznání vzorů v testovací množině. V grafu jsou červenou, respektive zelenou, linkou naznačeny časy učení on-line SOM, respektive BSOM.



Obrázek 21: Doba učení v závislosti na počtu Slave uzlů, GM-SOM, ručně psané číslice

8.3.2 Datová sada MNIST

Naměřené hodnoty pro datovou sadu MNIST a typ paralelizace GMSOM lze najít v tabulce 16. Postupným přidáváním výpočetních uzlů čas učení klesal až na hodnotu 1203 vteřin a to při 15 Slave uzlech. Oproti sekvenční verzi došlo k téměř pětinásobnému zrychlení vzhledem k on-line verzi, respektive k více než 1,5násobnému vzhledem k verzi BSOM. Další přidávání výpočetních uzlů již vedlo k prodlužování potřebné doby a to vzhledem k narůstající trénovací množině využité v posledním kroku učení algoritmu GM-SOM.

8.4 Datový paralelismus

Druhým ze čtyř implementovaných způsobů paralelizace je datový paralelismus. Na průběh učení může mít vliv velikost dávky, ovlivňující jak často dochází ke komunikaci a synchronizaci dat mezi řídicím a výpočetním uzlem, a také počet výpočetních uzlů. Proto byly provedena měření zaměřená právě na vliv těchto dvou parametrů.

8.4.1 Datová sada ručně psaných číslic

V tabulce 17 jsou uvedeny naměřené hodnoty pro postupně zvyšovanou velikost dávky. Z naměřených dat vyplývá, že s velikostí dávky rapidně klesá úspěšnost, v tomto případě se o naučení dá hovořit pouze pro velikost dávky 20, kdy byla úspěšnost rozpoznání vzorů testovací množiny 72,34 %. Pro dávky o velikosti více než 50 vzorů klesla úspěšnost na hodnoty kolem 10 %, což pro datovou sadu o 10 kategoriích odpovídá při vyvolání informace náhodnému výběru jedné

z nich. Při menší velikosti dávky dochází k častější komunikaci mezi uzly a to se projevilo na časové náročnosti učení. Časovou náročnost režie, která je navíc při učení na více uzlech oproti sekvenčnímu způsobu, se podařilo převážit až v případě, kdy velikost dávky dosahovala 150 vzorů a více.

Jak již bylo řečeno, vliv na výsledek může mít i měnící se počet výpočetních uzlů, proto bylo porovnáno chování při zvyšujícím se počtu uzlů pro učící dávky o velikosti 50, 100 a 200. Naměřené hodnoty lze vidět v tabulce 19. Pro počet uzlů menší než 7 byla úspěšnost srovnatelná se sekvenční verzí on-line SOM. V těchto případech však čas učení převyšoval délku učení sekvenčních verzí.

8.4.2 Datová sada MNIST

Při pohledu na naměřené hodnoty, kterých dosahoval algoritmus DP, lze usoudit, že pro datovou sadu o větším počtu vzorů, není tento způsob paralelizace SOM vhodný. V tabulce 18 jsou zaznamenány hodnoty pro měnící se velikost dávky a tabulka 20 obsahuje naměřené hodnoty pro měnící se počet Slave uzlů. Pro tento způsob byla úspěšnost srovnatelná se sekvenční verzí pouze v případě, kdy výpočet probíhal na 1 Slave uzlu.

8.5 Datový paralelismus, jedna iterace

I v tomto případě, podobně jako u GM-SOM, se celá trénovací množina rozprostře mezi jednotlivé výpočetní uzly. Proto je tedy velikost dávky určena právě počtem výpočetních uzlů.

8.5.1 Datová sada ručně psaných číslic

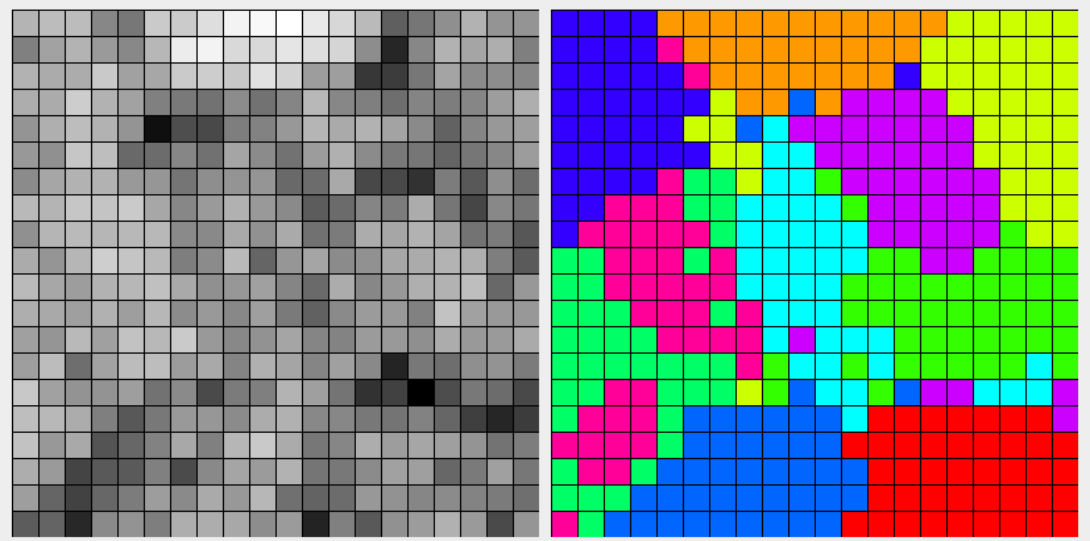
Naměřené hodnoty pro datovou sadu ručně psaných číslic v kombinaci s učícím algoritmem založeným na datovém paralelismu, kdy jsou jednotlivé dávky zpracovány pomocí on-line SOM na jednotlivých uzlech, jsou zobrazeny v tabulce 21. I v tomto případě úspěšnost značně klesala s přidaným počtem uzlů. Pokud byl při tomto způsobu využit pouze jeden Master a jeden Slave uzel, dosahovala úspěšnost naučení SOM srovnatelné výsledky s on-line sekvenčním algoritmem, prodloužení času zde pravděpodobně způsobila komunikace mezi uzly, která se v sekvenční verzi nevyskytuje. Po přidání dalších 2 Slave uzlů k výpočtu již úspěšnost spadla na 27,99 % a přidáváním dalších padala až na hodnotu kolem 10 %. Komunikace mezi Master a Slave uzlem je v tomto případě pouze při zahájení učení, a poté při odeslání výsledků zpět na Master uzel, tato vlastnost se pozitivně projevila na době učení.

8.5.2 Datová sada MNIST

Na základě dat v tabulce 22 se dá vyhodnotit, že tento způsob není pro paralelizaci učení Kohonenových map vhodný, podobně jak tomu bylo u předchozího způsobu. Jak v případě menšího datasetu, tak i v případě datové sady MNIST, se úspěšnost naučené sítě držela přes 90 % pouze v případě, kdy učení probíhalo na 1 Slave uzlu.

Na obrázcích 22 a 23 je vidět srovnání u-matrix a teplotní mapy pro sítě s výrazně odlišnou úspěšností. V prvním případě, obrázek 22,kdy byla velikost dávky 60000, byla úspěšnost rozpoznání vzorů 92,93, jednalo se tedy o poměrně dobře naučenou instanci sítě. Na teplotní mapě jsou viditelné shluky neuronů pro jednotlivé kategorie, kde neurony pro stejnou kategorii jsou označeny stejnou barvou. Vzdálenosti mezi neurony nejsou velmi velké, ale i přesto lze na u-matrix v tomto případě vidět tmavé linie oddělující jednotlivé shluky.

V případě, kdy byla velikost dávky 20000 a došlo k přidání dalších Slave uzlů, obrázek 23, je vidět, že nedošlo k utvoření většiny shluků a výstupní vrstva tedy nepokryla vstupní prostor dostatečně, aby bylo možné klasifikovat jednotlivé kategorie. Tomuto odpovídá i úspěšnost sítě pro rozpoznávání vzorů v testovací množině, kdy správě bylo určeno pouze 14,34 % vzorů.



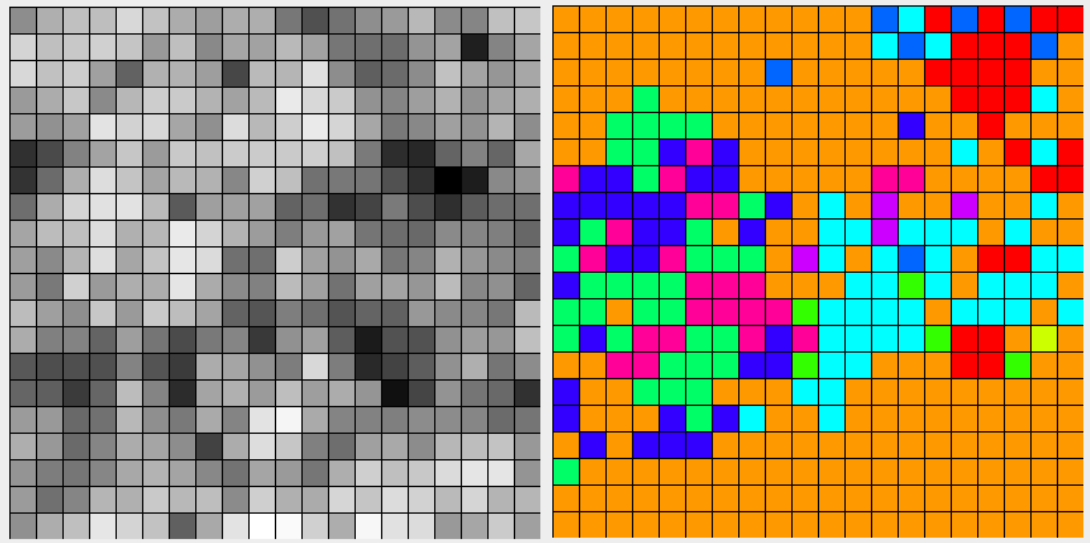
Obrázek 22: U-matrix a teplotní mapa pro DP s jednou iterací, MNIST, dávka 60000

8.6 Dávková SOM

Tento způsob paralelizace, stejně jako GM-SOM, je určen přímo pro neuronové sítě SOM, proto lze u těchto dvou způsobů očekávat nejlepší výsledky. Tento způsob, na rozdíl od GM-SOM, lze ovlivnit nastavením velikosti zpracovávané dávky a také počtem výpočetních uzlů, jelikož jsou zde tyto parametry na sobě nezávislé.

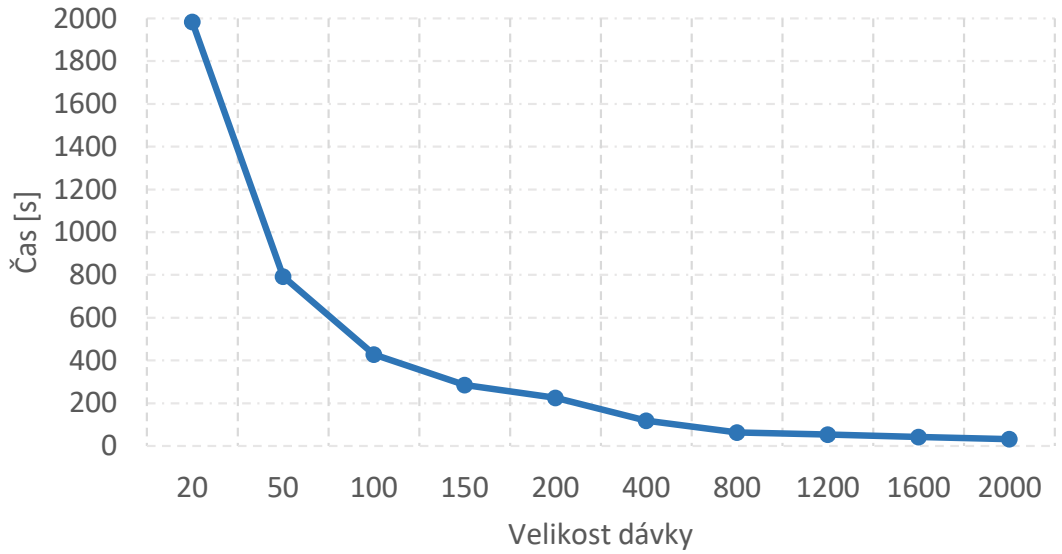
8.6.1 Datová sada ručně psaných číslic

Graf na obrázku 24, který je vytvořen na základě hodnot v tabulce 23, zobrazuje čas potřebný k naučení SOM v závislosti na velikosti dávky. S větší dávkou rapidně klesla doba učení a zároveň nedošlo k poklesu úspěšnosti naučené sítě. Téměř ve všech měřených případech, zobrazeny v tabulce 23, byla úspěšnost správného určení vzoru z testovací množiny přes 96 %. V dalších experimentech bylo vyzkoušeno přidání dalších výpočetních uzlů i zvýšení dávky až na hodnotu



Obrázek 23: U-matrix a teplotní mapa pro DP s jednou iterací, MNIST, dávka 20000

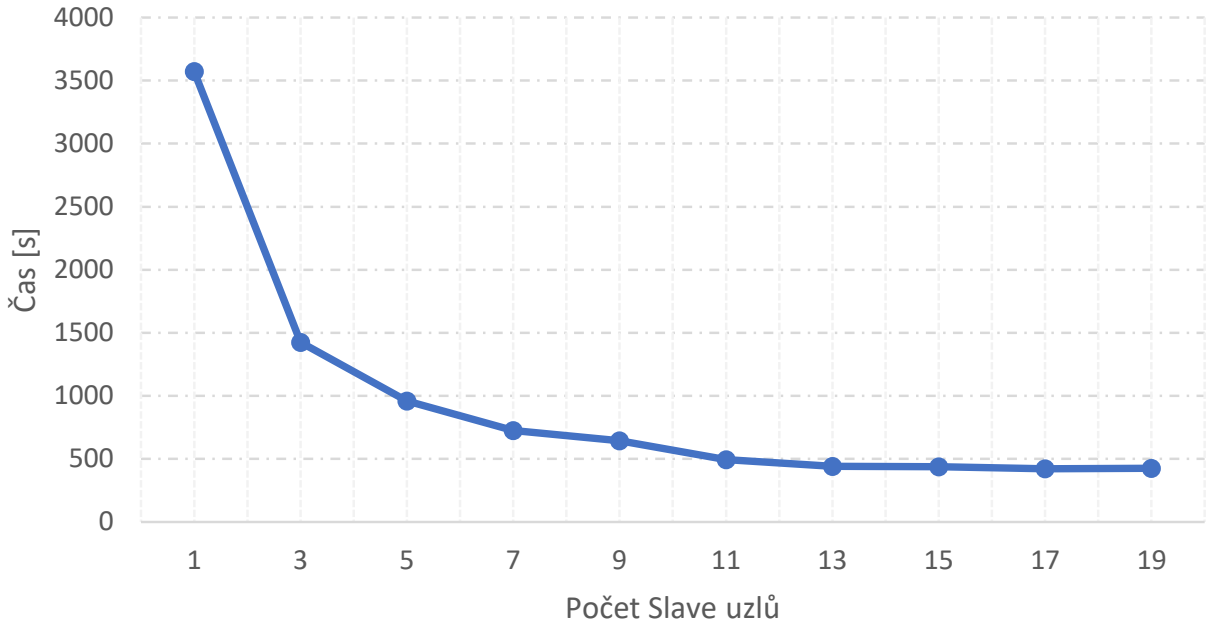
kdy pokrývala celou trénovací množinu, ale již nedošlo ke zlepšení času potřebného k učení. Nejlepší čas dosáhlo učení s 9 Slave uzly a velikostí dávky 2000. V tomto případě bylo učení oproti sekvenčnímu on-line učení více něž 3,5krát rychlejší se srovnatelnou úspěšností. Oproti sekvenční BSOM bylo zlepšení času o 28 vteřin a úspěšnost stoupla z 88,98 % na 96,27 %.



Obrázek 24: Doba učení v závislosti na velikosti dávky, BSOM, ručně psané číslice

8.6.2 Datová sada MNIST

Stejně jako pro dataset ručně psaných číslic dosahovala BSOM velmi dobrých výsledků, co se úspěšnosti týče i pro dataset MNIST. V tabulce 24, která zobrazuje vliv velikosti dávky, lze vidět, že s vzrůstající velikostí dávky docházelo k mírnému poklesu úspěšnosti a to o 2 % při porovnání dávky velikosti 200 a 20000. Od dávky velikosti 2000 již nedocházelo k výraznému zlepšení doby učení, a proto byla tato velikost dávky zvolena pro sledování chování v závislosti na počtu výpočetních uzlů. Naměřené hodnoty lze vidět v tabulce 25. V tomto případě se úspěšnost držela v rozmezí 91,1 - 92,32 procent. Dobu potřebnou k učení pro rozdílný počet Slave uzlů lze vidět na obrázku 25.



Obrázek 25: Doba učení v závislosti na počtu Slave uzlů, BSOM, MNIST

8.7 Vliv sousedství

Funkce sousedství zasahuje do průběhu učení v podobě počtu ovlivněných neuronů v okolí BMU, z tohoto důvodu je vhodné porovnat vliv tétu funkce na průběh učení SOM, jelikož se volba typu a tvaru použitého sousedství dá považovat za jeden ze vstupních parametrů. V rámci programu ParallelSOM je možné zvolit typ, tvar a také chování na konci mřížky. Charakteristiky všech možností již byly dříve popsány.

Porovnání bylo provedeno na datasetu ručně psaných číslic, kdy pro každý ze 4 způsobů paralelizace byl sledován vliv tvaru, typu a chování na konci mřížky. Tyto naměřené hodnoty byly následně zprůměrovány a jsou uvedeny v tabulce 26.

Tabulka 12: Vliv tvaru sousedství

Tvar	Čas [s]	Úspěšnost [%]	Správně	Špatně
Čtvercové	84	67,66	1216	581
Kruhové	93	77,66	1396	402
Šestiúhelníkové	91	78,24	1406	391
Trojúhelníkové	98	77,00	1384	413

8.7.1 Tvar

Tabulka 12 ukazuje vliv tvaru sousedství na dobu učení a úspěšnost. Nejvíce byla sousedstvím ovlivněna úspěšnost v případě čtvercového sousedství, kdy úspěšnost byla o přibližně 10 % nižší než v ostatních případech. Zbylé tvary jsou v úspěšnosti rozpoznávání vzorů téměř shodné, ale v případě trojúhelníkového tvaru byl čas učení mírně delší.

Tabulka 13: Vliv chování na konci mřížky neuronů

Na konci mřížky	Čas [s]	Úspěšnost [%]	Správně	Špatně
Pokračuje	91	72,63	1305	492
Nepokračuje	91	77,65	1395	402

8.7.2 Chování na konci mřížky

Druhý parametr pro funkci sousedství je chování na konci mřížky. Sousedství buď na konci mřížky končí nebo pokračuje, jak je zobrazeno na obrázku 6. V tabulce 13 lze vidět, že chování nemělo vliv na dobu učení. Co se však změnilo je úspěšnost ve prospěch chování, kdy sousedství na konci mřížky nepokračuje.

Tabulka 14: Vliv typu sousedství

	Čas [s]	Úspěšnost [%]	Správně	Špatně
$P = 1$	94	75,30	1353	444
$P \in \langle 0; 1 \rangle$	90	75,46	1356	441
$P \in \langle 0; 0,5 \rangle$	90	74,67	1342	455

8.7.3 Typ

Typ sousedství ovlivňuje způsob, jak moc jsou neurony v sousedství ovlivněny při učení. Na základě hodnot v tabulce 14 lze říci, že jednotlivé nastavení tohoto parametru se téměř neliší.

Úspěšnost se lišila pouze v rozmězí jednoho procenta. Čas učení se lišil pouze v případě, kdy jsou všechny neurony v sousedství ovlivněny stejně jako BMU, a byl o 4 vteřiny delší než v ostatních případech.

9 Závěr

Tato práce se zabývala samoorganizujícími se mapami a způsoby paralelizace učení pro tyto sítě. V rámci práce byly vysvětleny obecné principy neuronových sítí. Dále zde byly detailně rozebrány samoorganizující se mapy, jejich vlastnosti a způsob učení.

Další část byla zaměřena na problematiku paralelizace tohoto typu neuronových sítí. Byly nastíněny obecné principy využívané při paralelizaci neuronových sítí. Práce se věnuje především Kohonenovým mapám, proto byly rozebrány také specifické přístupy paralelizace právě pro tento typ sítí.

Cílem práce bylo rozšíření stávajícího programu Modeler neuronových sítí o modul Kohonenových map. Tento modul umožňuje vytvořit, naučit a dále vizualizovat průběh učení či vnitřní strukturu sítě. Součástí modulu je také podpora pro celkem 4 tvary a 3 typy chování funkce sousedství, která je využívaná v průběhu učení Kohonenových map. Při návrhu modulu byl kladen důraz na jednoduchou a srozumitelnou vizualizaci chování sítě, tak aby byl přínosem ve výuce předmětu Neuronové sítě.

Další z cílů práce byla možnost spouštět učení Kohonenových map na distribuovaných výpočetních uzlech, k tomuto účelu byl navrhnut a implementován další program. Program pro paralelní učení Kohonenových map spolupracuje s Modelerem neuronových sítí, kdy uživatel může v modeleru navrhnout strukturu sítě a nastavit učící parametry. Pomocí konfiguračního souboru je poté takto navržená síť naučena programem pro paralelizaci, který je možno spouštět na distribuovaných výpočetních uzlech. Výstup z druhého programu v podobě naučené sítě je možné dále načíst do programu Modeler a zde dále analyzovat tuto sít.

V poslední kapitole byla provedena analýza jednotlivých způsobů paralelizace. K tomuto účelu byly použity dvě datové sady. Pro jednotlivé implementované způsoby paralelizace bylo zkoumáno na těchto sadách chování při výpočtech na výpočetních uzlech. Výpočty byly prováděny v prostředí superpočítače Salomon.

Při analýze se ukázalo, že obecné přístupy paralelizace neuronových sítí, kdy dochází k synchronizaci dat mezi uzly pomocí gradientů vah, nejsou v případě Kohonenových map velmi vhodné, jelikož docházelo k rychlému poklesu úspěšnosti v závislosti na přidávání výpočetních uzlů. Při snižování velikosti učící dávky se úspěšnost držela, pro menší datovou sadu, na srovnatelné hranici se sekvenční verzí, častá komunikace však v tomto případě způsobila prodloužení času výpočtu a vhodnější volbou se tak ukázala sekvenční varianta bez paralelizace.

V případě způsobů paralelizace navržených speciálně pro Kohonenovy mapy již byla situace rozdílná, úspěšnost naučených sítí byla v tomto případě téměř vždy srovnatelná se sekvenční verzí. U algoritmu GM-SOM podařilo dosáhnout pětinásobného zrychlení oproti klasické sekvenční verzi učení a více než 1,5násobnému zrychlení oproti dávkové variantě učení bez paralelizace, v případě datové sady MNIST.

V případě paralelizace dávkového způsobu učení Kohonenových map, byly výsledky ještě lepší než v případě GM-SOM. Tento způsob se na základě měření zdá být nejlepší možnou

volbou z implementovaných způsobů. V tomto případě nedocházelo k propadu úspěšnosti ani k prodlužování času potřebnému k učení na základě přidávání výpočetních uzlů.

Velkou roli na samoorganizaci a shlukování neuronů Kohonenových map má funkce sousedství využívaná v průběhu učení. Proto byla provedena také měření, kdy byl sledován vliv tvaru a typu sousedství na úspěšnost a čas učení sítě. V tomto případě bylo zjištěno, že ze čtyř implementovaných způsobů mělo nejnižší úspěšnost sousedství tvaru čtvercového, ostatní tvary byly srovnatelné. V případě chování funkce sousedství na konci mřížky neuronů lze, na základě naměřených dat, doporučit, aby sousedství končilo na konci mřížky a nepokračovalo.

Literatura

[1] Aeron. [online], Dostupné z https://github.com/real-logic/aeron, Naposledy navštíveno 12. 4. 2020.

[2] Yiu-ming Cheung and Lap-tak Law. Rival-model penalized self-organizing map. *IEEE Transactions on Neural Networks*, 18(1):289–295, 2007.

[3] George Dahl, Alan McAvinney, Tia Newhall, et al. Parallelizing neural network training for cluster systems. In *Proceedings of the IASTED international conference on parallel and distributed computing and networks*, pages 220–225. ACTA Press, 2008.

[4] IT4Innovations Documentation. [online], Dostupné z https://docs.it4i.cz/salomon/introduction/, Naposledy navštíveno 1. 5. 2020.

[5] Mark J Embrechts and Fabio Arciniegas. Neural networks for text-to-speech phoneme recognition. In *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, volume 5, pages 3582–3587. IEEE, 2000.

[6] Petr Gajdoš and Pavel Moravec. Two-step modified som for parallel calculation. In *Proceedings of DATESO*, pages 13–21. Citeseer, 2010.

[7] GIT. [online], Dostupné z https://git-scm.com/, Naposledy navštíveno 10. 4. 2020.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[9] Kevin Gurney. *An introduction to neural networks*. CRC press, 2014.

[10] Simon S Haykin et al. Neural networks and learning machines/simon haykin., 2009.

[11] Eclipse IDE. [online], Dostupné z https://www.eclipse.org/, Naposledy navštíveno 10. 4. 2020.

[12] jMonkeyEngine. [online], Dostupné z https://jmonkeyengine.org/, Naposledy navštíveno 5. 4. 2020.

[13] Richard D. Lawrence, George S. Almasi, and Holly E. Rushmeier. A scalable parallel algorithm for self-organizing maps with applications to sparse data mining problems. *Data Mining and Knowledge Discovery*, 3(2):171–195, 1999.

[14] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[15] THE MNIST DATABASE of handwritten digits. [online], Dostupné z http://yann.lecun.com/exdb/mnist/, Naposledy navštíveno 10. 4. 2020.

[16] Optical Recognition of Handwritten Digits Data Set. [online], Dostupné z http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits, Naposledy navštíveno 10. 4. 2020.

[17] Vojtěch Pustówka. Rozšiřující knihovna pro modeler neuronovỳch sítí. 2019.

[18] Constantino Carlos Reyes-Aldasoro. Image segmentation with kohonen neural network self-organising maps. In *International Conference on Telecommunications ICT*. Citeseer, 2000.

[19] Iris Data Set. [online], Dostupné z http://archive.ics.uci.edu/ml/datasets/Iris/, Naposledy navštíveno 8. 4. 2020.

[20] Christopher J Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

[21] Bruno Silva and Nuno Marques. A hybrid parallel som algorithm for large maps in data-mining. *New Trends in Artificial Intelligence*, 2007.

[22] Pavel Stefanovic and Olga Kurasova. Visual analysis of self-organizing maps. *Nonlinear Analysis: Modelling and Control*, 16(4):488–504, 2011.

[23] A VOJÁČEK. Samoučící se neuronová síť'. *Datum publikování*, pages 05–14, 2006.

[24] Eva Volná. Neuronové sítě 1. *Ostrava: Ostravská univerzita v Ostravě. Vydání: druhé*, 2008.

[25] Ivo VONDRÁK. Neuronové sítě. *Fakulta elektrotechniky a informatiky VŠB. Učební text. Ostrava*, 2009.

[26] Ivan ZELINKA. Umělá inteligence. 2. vyd. 127 s. *Zlín: Univerzita Tomáše Bati ve Zlíně.*, 2005.

A Naměřené hodnoty

Tabulka 15: Naměřené hodnoty pro algoritmus GM-SOM a dataset ručně psaných číslic

Algoritmus	Dávka	Počet uzlů	Instancí		Čas [s]	Úspěšnost [%]	Správně	Špatně
			Master	Slave				
GM-SOM	3900	2	1	1	118	95,33	1713	84
GM-SOM	1300	2	1	3	62	94,99	1707	90
GM-SOM	780	2	1	5	59	94,21	1693	104
GM-SOM	558	2	1	7	69	95,05	1708	89
GM-SOM	434	2	1	9	80	95,27	1712	85
GM-SOM	355	2	1	11	89	95,10	1709	88
GM-SOM	300	2	1	13	106	95,21	1711	86
GM-SOM	260	2	1	15	118	95,10	1709	88
GM-SOM	230	2	1	17	129	95,33	1713	84
GM-SOM	206	2	1	19	141	95,10	1709	88

Tabulka 16: Naměřené hodnoty pro algoritmus GM-SOM a dataset MNIST

Algoritmus	Dávka	Počet uzlů	Instancí		Čas [s]	Úspěšnost [%]	Správně	Špatně
			Master	Slave				
GM-SOM	60000	1	1	1	6808	89,23	8923	1077
GM-SOM	20000	1	1	3	2474	90,49	9049	951
GM-SOM	12000	1	1	5	1617	90,40	9040	960
GM-SOM	8572	1	1	7	1433	90,52	9052	948
GM-SOM	6667	1	1	9	1367	89,43	8943	1057
GM-SOM	5455	2	1	11	1216	90,14	9014	986
GM-SOM	4616	2	1	13	1261	89,89	8989	1011
GM-SOM	4000	2	1	15	1203	89,66	8966	1034
GM-SOM	3530	2	1	17	1916	90,31	9031	969
GM-SOM	3158	2	1	19	1925	90,23	9023	977

Tabulka 17: Vliv velikosti dávky na datový paralelismus a dataset ručně psaných číslic

Algoritmus	Dávka	Počet uzlů	Instancí		Čas [s]	Úspěšnost [%]	Správně	Špatně
			Master	Slave				
DP	20	2	1	9	548	72,34	1300	497
DP	50	2	1	9	230	9,91	178	1619
DP	100	2	1	9	134	9,91	178	1619
DP	150	2	1	9	92	9,91	178	1619
DP	200	2	1	9	76	9,91	178	1619
DP	400	2	1	9	49	10,02	180	1617
DP	800	2	1	9	35	9,96	179	1618
DP	1200	2	1	9	33	10,18	183	1614
DP	1600	2	1	9	31	9,91	178	1619
DP	2000	2	1	9	27	10,02	180	1617

Tabulka 18: Vliv velikosti dávky na datový paralelismus a dataset MNIST

Algoritmus	Dávka	Počet uzlů	Instancí		Čas [s]	Úspěšnost [%]	Správně	Špatně
			Master	Slave				
DP	200	1	1	9	1971	8,92	892	9108
DP	500	1	1	9	1123	8,92	892	9108
DP	1000	1	1	9	994	9,74	974	9026
DP	1500	1	1	9	972	9,80	980	9020
DP	2000	1	1	9	955	9,82	982	9018
DP	3000	1	1	9	947	10,09	1009	8991
DP	5000	1	1	9	955	10,28	1028	8972
DP	7000	1	1	9	951	9,82	982	9018
DP	10000	1	1	9	942	10,32	1032	8968
DP	20000	1	1	9	949	22,82	2282	7718

Tabulka 19: Vliv počtu výpočetních uzlů na datový paralelismus a dataset ručně psaných číslic

Algoritmus	Dávka	Počet uzlů	Instancí		Čas [s]	Úspěšnost [%]	Správně	Špatně
			Master	Slave				
DP	50	2	1	1	4485	95,54	1717	80
DP	50	2	1	3	1012	95,94	1724	73
DP	50	2	1	5	552	96,16	1728	69
DP	50	2	1	7	252	89,59	1610	187
DP	50	2	1	9	230	9,91	178	1619
DP	100	2	1	1	2494	95,88	1723	74
DP	100	2	1	3	454	96,61	1736	61
DP	100	2	1	5	274	96,44	1733	64
DP	100	2	1	7	138	9,91	178	1619
DP	100	2	1	9	129	9,91	178	1619
DP	200	2	1	1	1766	95,60	1718	79
DP	200	2	1	3	313	96,88	1741	56
DP	200	2	1	5	179	95,83	1722	75
DP	200	2	1	7	82	10,07	181	1616
DP	200	2	1	9	76	9,91	178	1619

Tabulka 20: Vliv počtu výpočetních uzlů na datový paralelismus a dataset MNIST

Algoritmus	Dávka	Počet uzlů	Instancí		Čas [s]	Úspěšnost [%]	Správně	Špatně
			Master	Slave				
DP	2000	1	1	1	6402	92,5	9250	750
DP	2000	1	1	3	2189	11,35	1135	8865
DP	2000	1	1	5	1444	9,58	958	9042
DP	2000	1	1	7	1123	9,80	980	9020
DP	2000	1	1	9	955	9,82	982	9018
DP	2000	2	1	11	724	9,80	980	9020
DP	2000	2	1	13	650	9,82	982	9018
DP	2000	2	1	15	591	9,74	974	9026
DP	2000	2	1	17	565	9,74	974	9026
DP	2000	2	1	19	530	9,80	980	9020

Tabulka 21: Naměřené hodnoty pro algoritmus DP s jednou iterací a dataset ručně psaných číslic

Algoritmus	Dávka	Počet uzlů	Instancí		Čas [s]	Úspěšnost [%]	Správně	Špatně
			Master	Slave				
DP, jedna iterace	3900	2	1	1	117	96,49	1734	63
DP, jedna iterace	1300	2	1	3	45	27,99	503	1294
DP, jedna iterace	780	2	1	5	31	19,20	345	1452
DP, jedna iterace	558	2	1	7	26	10,13	182	1615
DP, jedna iterace	434	2	1	9	24	10,13	182	1615
DP, jedna iterace	355	2	1	11	24	10,13	182	1615
DP, jedna iterace	300	2	1	13	24	10,13	182	1615
DP, jedna iterace	260	2	1	15	25	10,13	182	1615
DP, jedna iterace	230	2	1	17	28	10,13	182	1615
DP, jedna iterace	206	2	1	19	30	10,13	182	1615

Tabulka 22: Naměřené hodnoty pro algoritmus DP s jednou iterací a dataset MNIST

Algoritmus	Dávka	Počet uzlů	Instancí		Čas [s]	Úspěšnost [%]	Správně	Špatně
			Master	Slave				
DP, jedna iterace	60000	1	1	1	6720	92,93	9293	707
DP, jedna iterace	20000	1	1	3	2175	14,34	1434	8566
DP, jedna iterace	12000	1	1	5	1472	11,35	1135	8865
DP, jedna iterace	8572	1	1	7	1116	11,35	1135	8865
DP, jedna iterace	6667	1	1	9	988	11,35	1135	8865
DP, jedna iterace	5455	2	1	11	726	11,35	1135	8865
DP, jedna iterace	4616	2	1	13	607	11,35	1135	8865
DP, jedna iterace	4000	2	1	15	663	11,35	1135	8865
DP, jedna iterace	3530	2	1	17	607	11,35	1135	8865
DP, jedna iterace	3158	2	1	19	575	11,35	1135	8865

Tabulka 23: Vliv velikosti dávky na BSOM a dataset ručně psaných číslic

Algoritmus	Dávka	Počet uzlů	Instancí		Čas [s]	Úspěšnost [%]	Správně	Špatně
			Master	Slave				
BSOM	20	2	1	9	1984	96,05	1726	71
BSOM	50	2	1	9	793	96,61	1736	61
BSOM	100	2	1	9	428	96,38	1732	65
BSOM	150	2	1	9	286	96,77	1739	58
BSOM	200	2	1	9	225	96,77	1739	58
BSOM	400	2	1	9	118	96,88	1741	56
BSOM	800	2	1	9	63	96,83	1740	57
BSOM	1200	2	1	9	53	96,66	1737	60
BSOM	1600	2	1	9	42	95,88	1723	74
BSOM	2000	2	1	9	32	96,27	1730	67

Tabulka 24: Vliv velikosti dávky na BSOM a dataset MNIST

Algoritmus	Dávka	Počet uzlů	Instancí		Čas [s]	Úspěšnost [%]	Správně	Špatně
			Master	Slave				
BSOM	200	1	1	9	2792	92,32	9232	768
BSOM	500	1	1	9	1232	91,77	9177	823
BSOM	1000	1	1	9	762	92,37	9237	763
BSOM	1500	1	1	9	673	91,71	9171	829
BSOM	2000	1	1	9	645	92,06	9206	794
BSOM	3000	1	1	9	649	91,58	9158	842
BSOM	5000	1	1	9	643	90,98	9098	902
BSOM	7000	1	1	9	640	91,70	9170	830
BSOM	10000	1	1	9	639	90,74	9074	926
BSOM	20000	1	1	9	628	90,19	9019	981

Tabulka 25: Vliv počtu výpočetních uzlů na BSOM a dataset MNIST

Algoritmus	Dávka	Počet uzlů	Instancí		Čas [s]	Úspěšnost [%]	Správně	Špatně
			Master	Slave				
BSOM	2000	1	1	1	3572	92,10	9210	790
BSOM	2000	1	1	3	1425	92,32	9232	768
BSOM	2000	1	1	5	961	92,10	9210	790
BSOM	2000	1	1	7	727	91,88	9188	812
BSOM	2000	1	1	9	645	92,06	9206	794
BSOM	2000	2	1	11	496	91,10	9110	890
BSOM	2000	2	1	13	443	91,23	9123	877
BSOM	2000	2	1	15	439	91,52	9152	848
BSOM	2000	2	1	17	422	91,37	9137	863
BSOM	2000	2	1	19	425	91,65	9165	835

Tabulka 26: Vliv sousedství

Tvar	Typ	Pokračuje na konci mřížky	Čas [s]	Úspěšnost [%]	Správně	Špatně
Čtvercové	$P = 1$	NE	92	79,12	1422	375
Čtvercové	$P \in \langle 0; 1 \rangle$	NE	89	78,93	1418	379
Čtvercové	$P \in \langle 0; 0,5 \rangle$	NE	88	78,13	1404	393
Čtvercové	$P = 1$	ANO	77	56,96	1024	774
Čtvercové	$P \in \langle 0; 1 \rangle$	ANO	77	57,12	1027	771
Čtvercové	$P \in \langle 0; 0,5 \rangle$	ANO	79	55,73	1002	796
Kruhové	$P = 1$	NE	94	78,31	1407	390
Kruhové	$P \in \langle 0; 1 \rangle$	NE	91	78,23	1406	391
Kruhové	$P \in \langle 0; 0,5 \rangle$	NE	89	78,35	1408	389
Kruhové	$P = 1$	ANO	99	76,43	1374	424
Kruhové	$P \in \langle 0; 1 \rangle$	ANO	94	76,75	1379	418
Kruhové	$P \in \langle 0; 0,5 \rangle$	ANO	93	77,87	1399	398
Šestiúhelníkové	$P = 1$	NE	93	79,66	1432	366
Šestiúhelníkové	$P \in \langle 0; 1 \rangle$	NE	87	77,10	1386	412
Šestiúhelníkové	$P \in \langle 0; 0,5 \rangle$	NE	89	75,61	1359	438
Šestiúhelníkové	$P = 1$	ANO	96	79,75	1433	364
Šestiúhelníkové	$P \in \langle 0; 1 \rangle$	ANO	90	79,86	1435	362
Šestiúhelníkové	$P \in \langle 0; 0,5 \rangle$	ANO	92	77,49	1393	405
Trojúhelníkové	$P = 1$	NE	97	75,71	1361	437
Trojúhelníkové	$P \in \langle 0; 1 \rangle$	NE	95	77,31	1389	408
Trojúhelníkové	$P \in \langle 0; 0,5 \rangle$	NE	94	75,40	1355	442
Trojúhelníkové	$P = 1$	ANO	104	76,44	1374	424
Trojúhelníkové	$P \in \langle 0; 1 \rangle$	ANO	100	78,38	1409	389
Trojúhelníkové	$P \in \langle 0; 0,5 \rangle$	ANO	99	78,79	1416	381